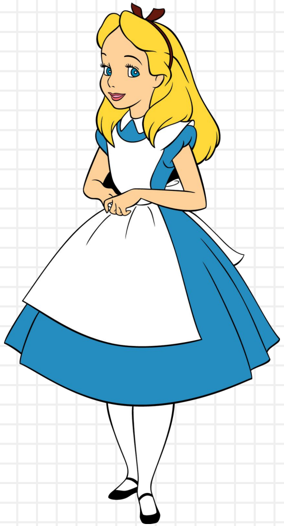
# Verifiable Computation over Encrypted Data: SNARKs and more

29 March 2022 - FHE.org


Anca Nitulescu

**Protocol Labs**

# Storage Delegation



Client

**Client:**

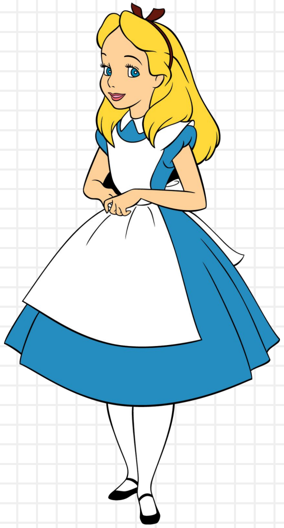
- ✘ limited storage
- ✘ minimal operating system
- ✘ limited computational power

**Cloud Service**

- ✘ **provides storage**

Server

# Storage Delegation



Client

## Client:

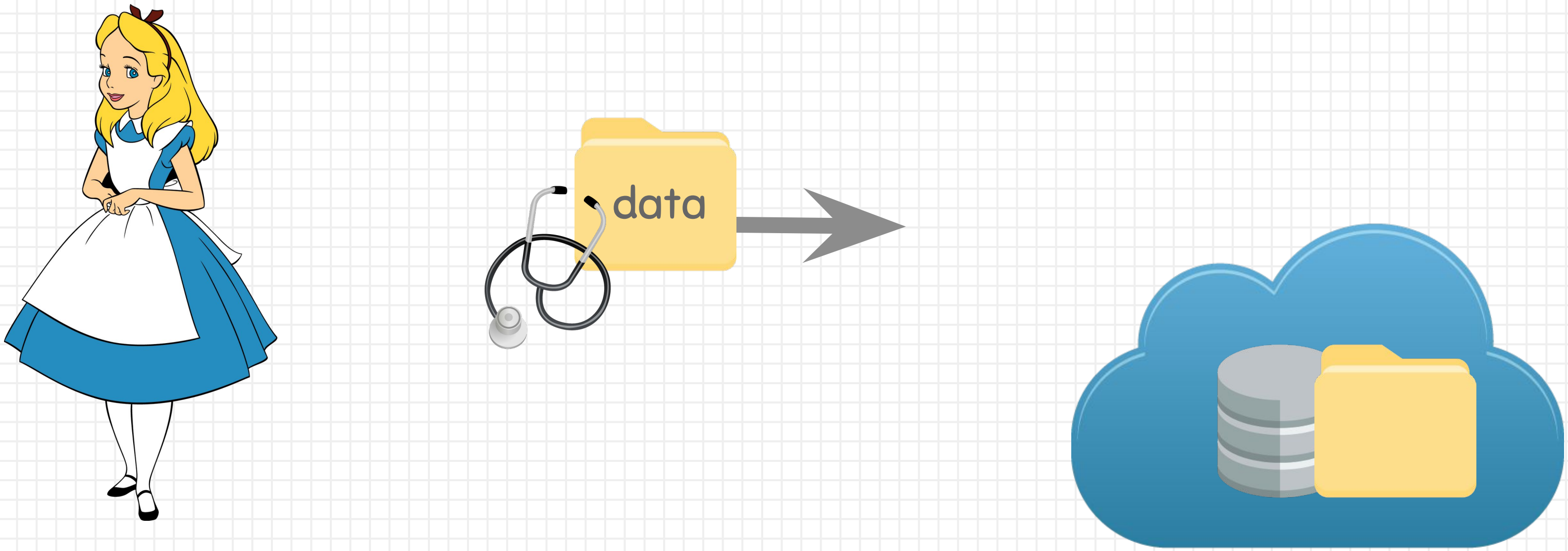
- ✘ limited storage
- ✘ minimal operating system
- ✘ limited computational power

## Cloud Service

- ✘ **provides storage**
- ✘ **computing power**
- ✘ network
- ✘ software

Server

# Storage Delegation

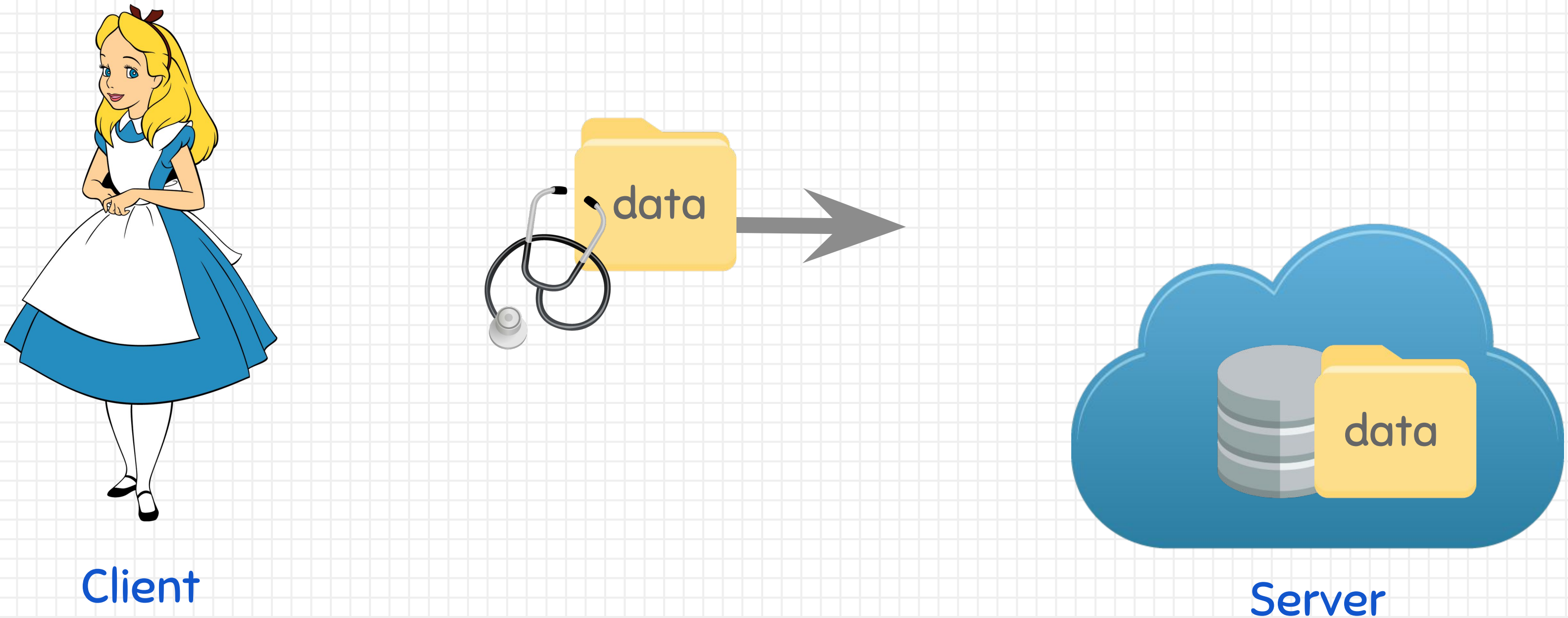



Client

Server

User delegates its personal data

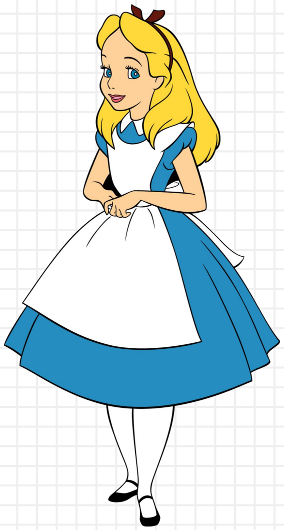
# Storage Delegation




Server stores the data

# Computation Delegation

$f(data)=y$



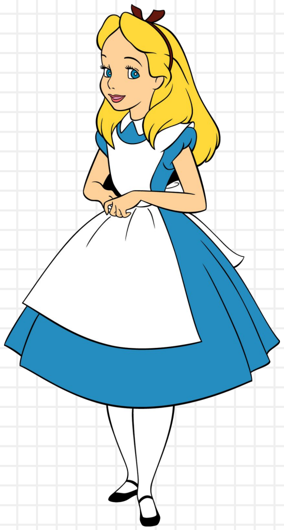
Client


Server

Server computes on the data

# Computation Delegation

f(data)=y



Client

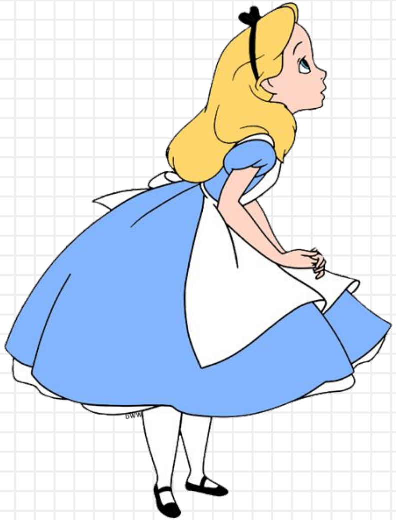
Claims that

f(data)=y


Server

User receives results

# What can go wrong? Data is exposed

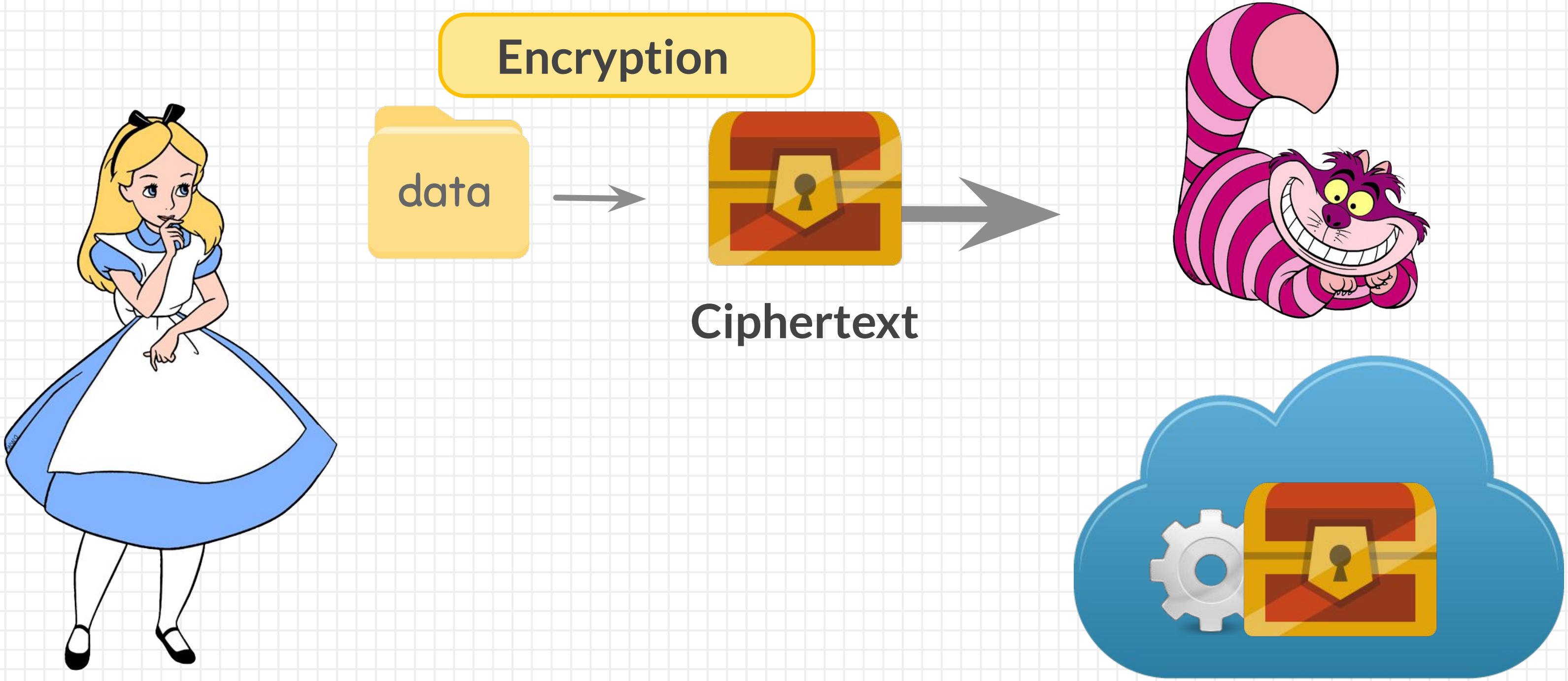



Client

Server

User's confidential data is exposed

# FHE: Solution for Privacy of Inputs



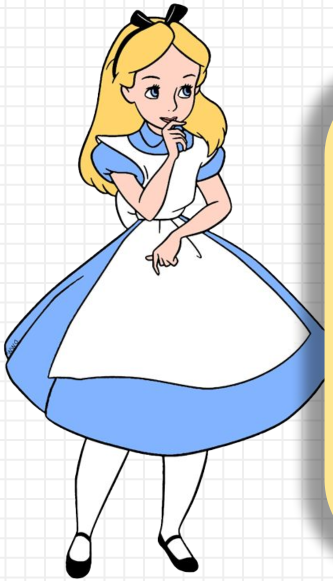

Client

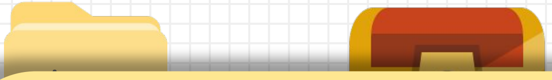
Server

User encrypts her data

# FHE: Solution for Privacy of Inputs

Encryption

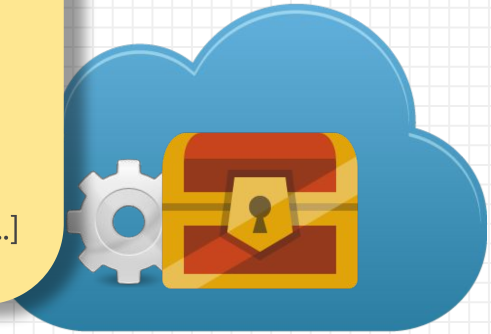




### Homomorphic Encryption

- ✘ Privacy of inputs
- ✘ Malleability of data
- ✘ Privacy of output

[Gen09, BGV12, GSW13, TFHE (CGGI16), CKKS17...]

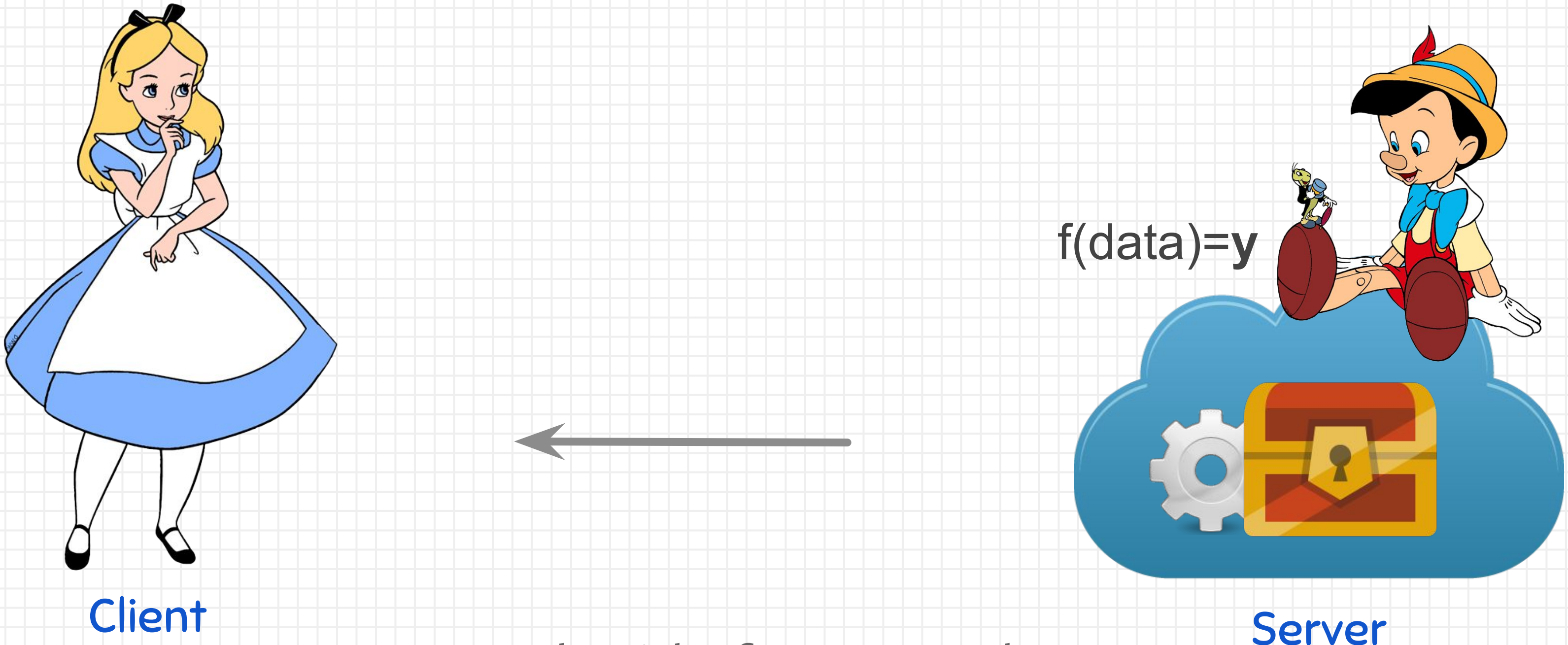


Client

User encrypts her data

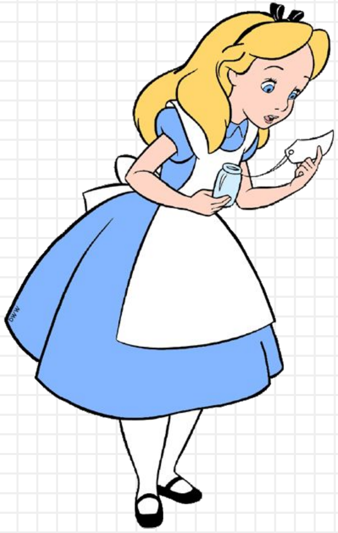
Server

# What can go wrong? Dishonest Server



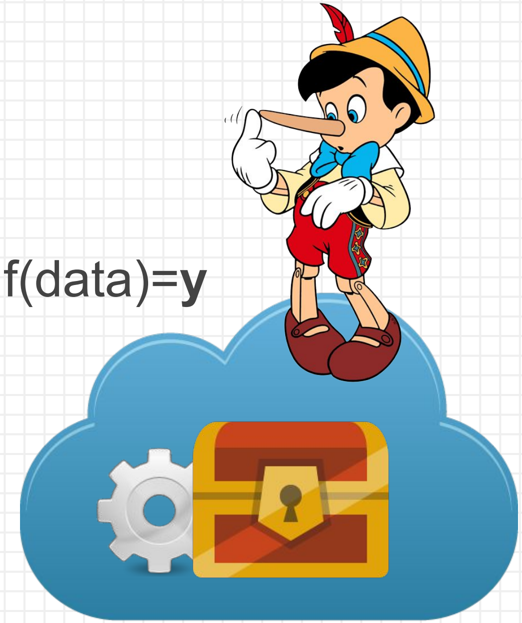

User runs the risk of a corrupted server

# What can go wrong? Dishonest Server



Client


f(data)=**y**



f(data)=**y’**

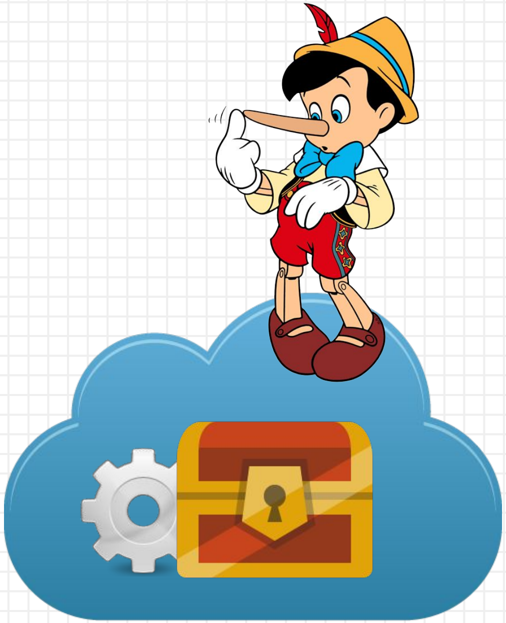
Server

Server sends incorrect results

# What can go wrong? Dishonest Server


Client

f(data)=y’

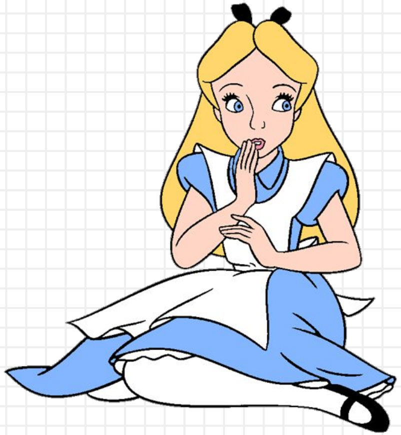


Server

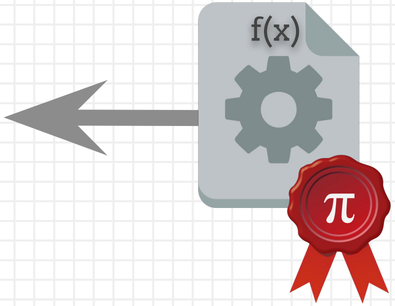
Server sends incorrect results

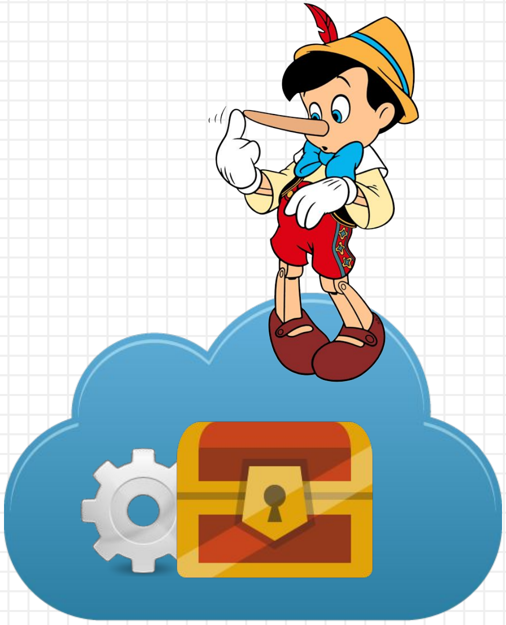
# SNARK: Solution for integrity of results

**Verifiable Computation**



Client

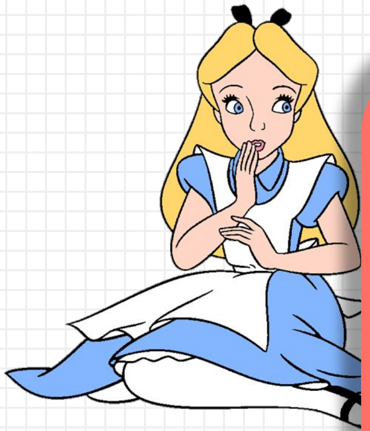


Server

User asks for a proof

# SNARK: Solution for integrity of results





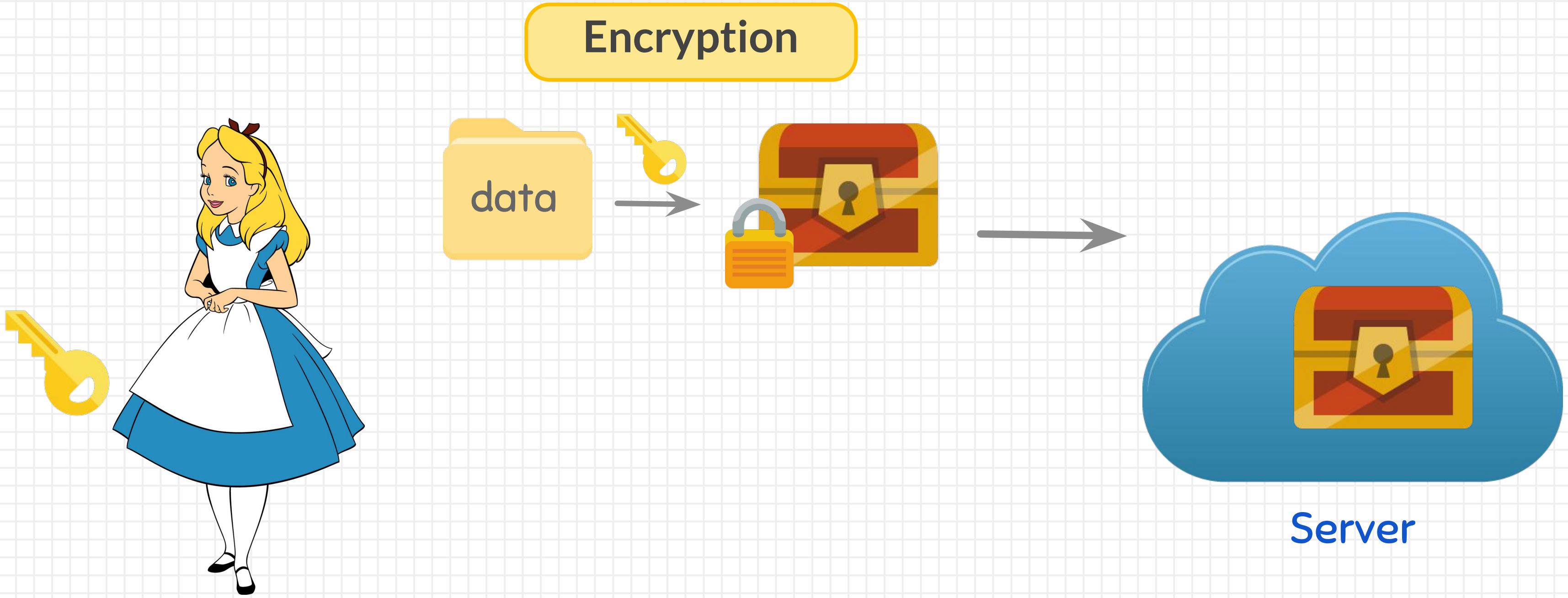
**Verifiable Computation**

SNARKs

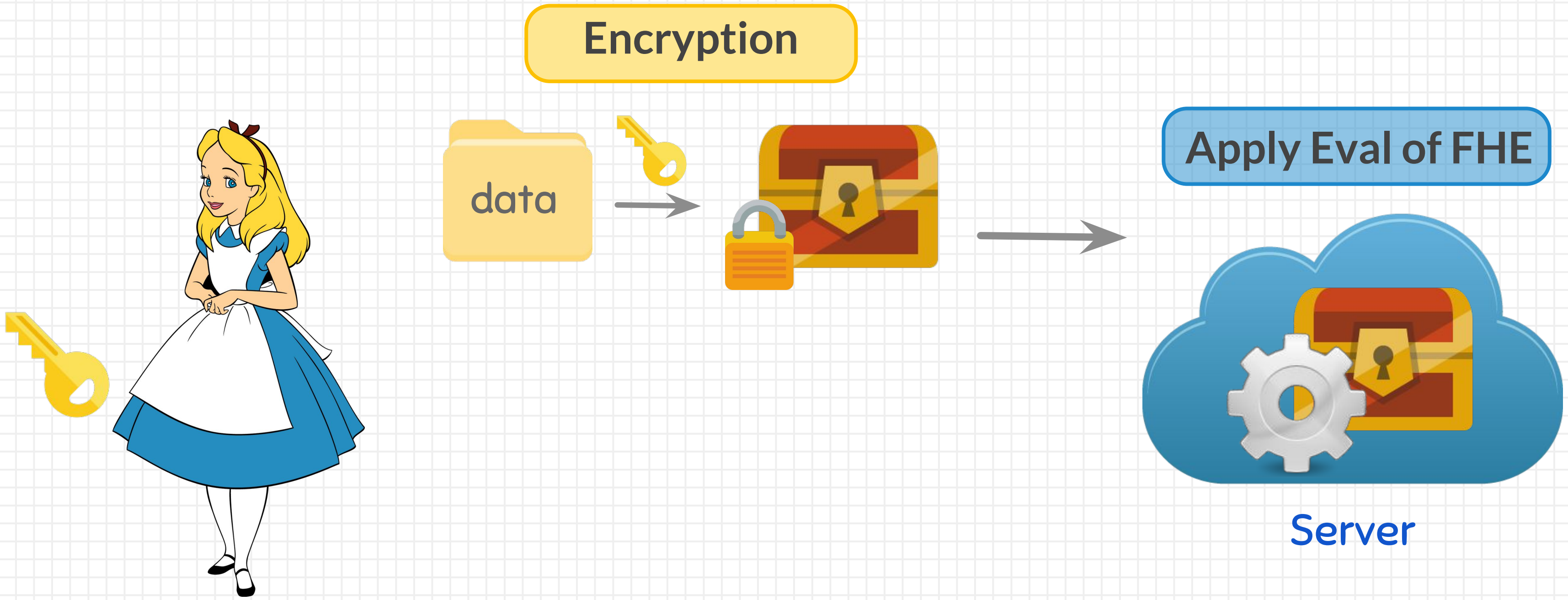
- ✘ Proof is succinct
- ✘ Minimal interaction
- ✘ Client verifies efficiently

[GGP10, GGPR13, PHGR13, Gro16, BBC+18...]

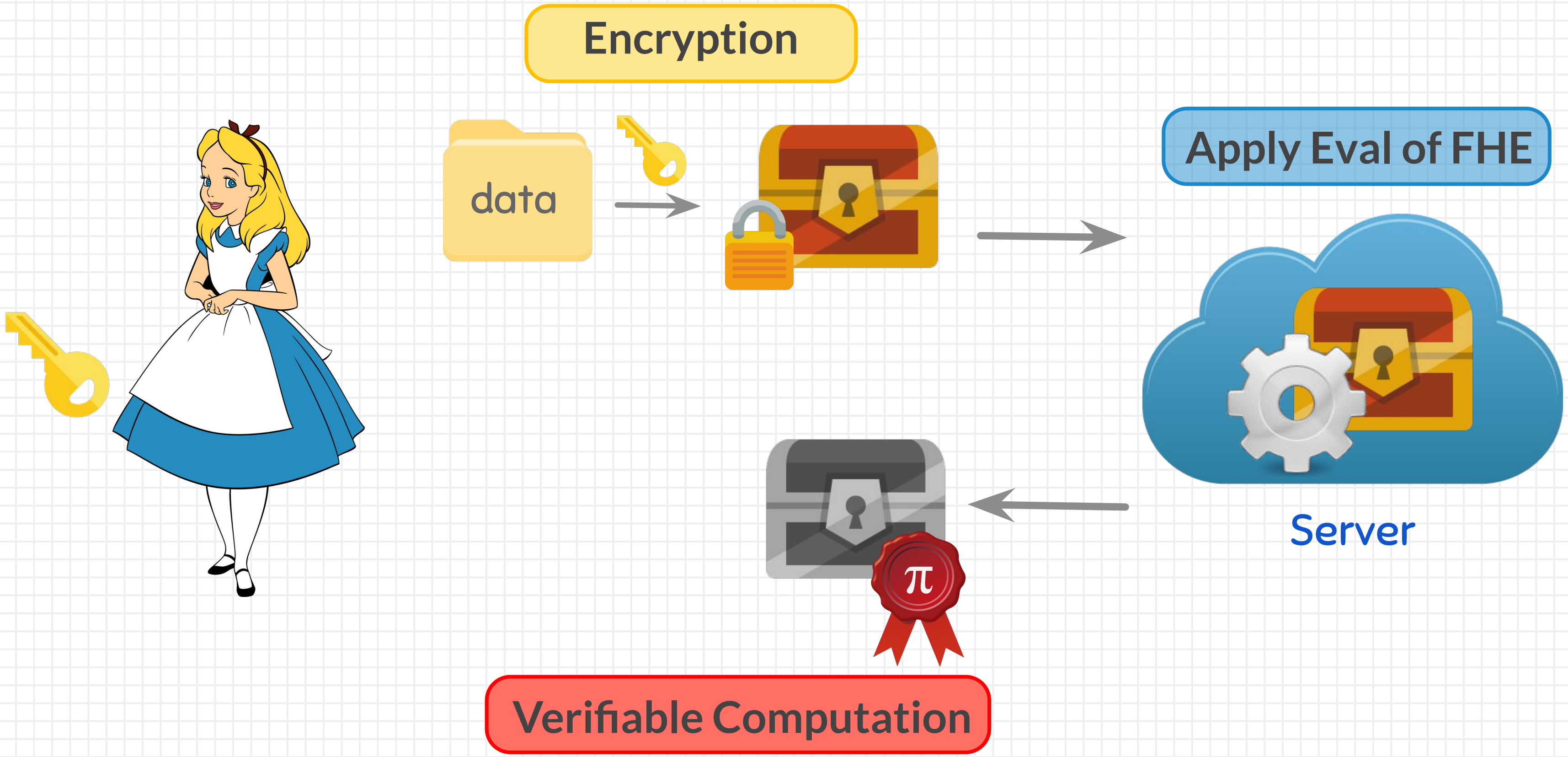
# Full Solution: Verifiable Computation on Encrypted Data



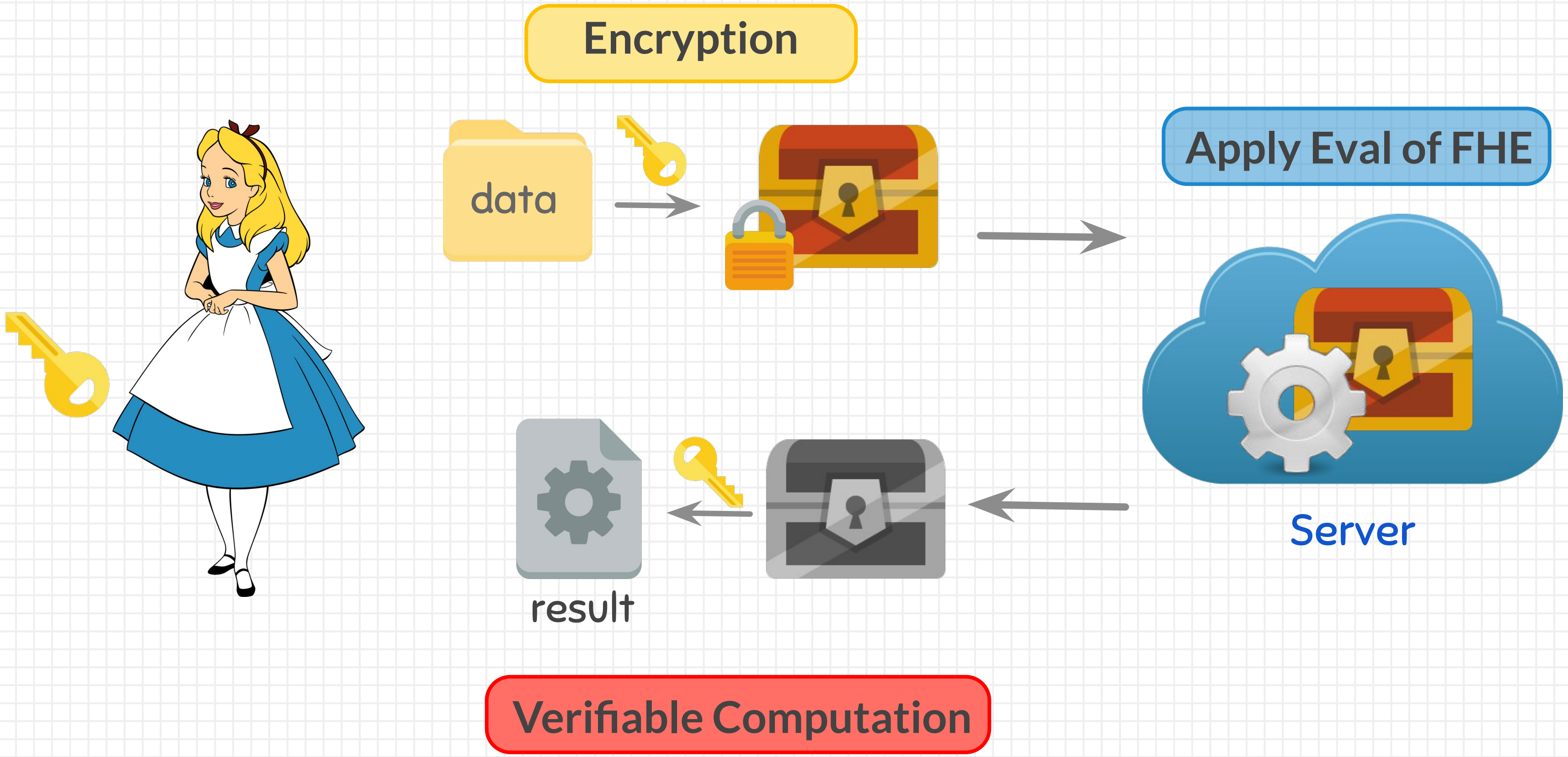
# Full Solution: Verifiable Computation on Encrypted Data



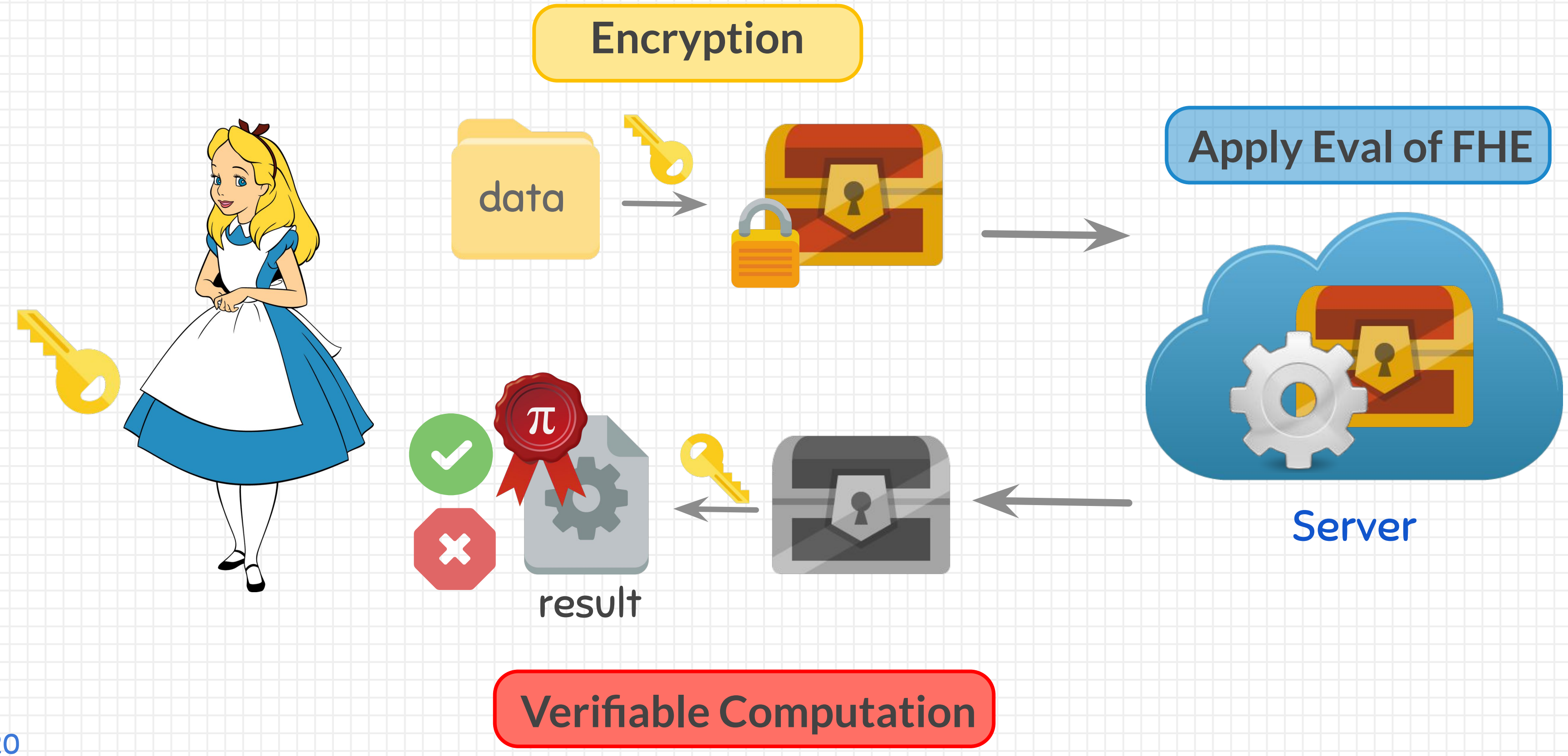
# Full Solution: Verifiable Computation on Encrypted Data



# Full Solution: Verifiable Computation on Encrypted Data



# Full Solution: Verifiable Computation on Encrypted Data



# Privacy-preserving Verifiable Computation

**Boosted SNARKs with data privacy for the inputs and outputs**

[PKC:FNP20] *Boosting Verifiable Computation on Encrypted Data*

Dario Fiore, **Anca Nitulescu**, David Pointcheval

# Privacy-preserving Verifiable Computation

**Boosted SNARKs with data privacy for the inputs and outputs**

[PKC:FNP20] *Boosting Verifiable Computation on Encrypted Data*
Dario Fiore, **Anca Nitulescu**, David Pointcheval

**Short-sighted SNARKs for Private Polynomial Evaluation and PSI**

[EP:2021/1291] *MyOPE: Malicious securitY for Oblivious Polynomial Evaluation*
Malika Izabachène, **Anca Nitulescu**, Paola de Perthuis, David Pointcheval

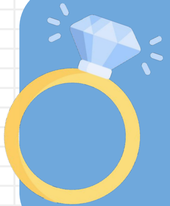
# Privacy-preserving Verifiable Computation

**Boosted SNARKs with data privacy for the inputs and outputs**

[PKC:FNP20] *Boosting Verifiable Computation on Encrypted Data*
Dario Fiore, **Anca Nitulescu**, David Pointcheval

**Short-sighted SNARKs for Private Polynomial Evaluation and PSI**

[EP:2021/1291] *MyOPE: Malicious securitY for Oblivious Polynomial Evaluation*
Malika Izabachène, **Anca Nitulescu**, Paola de Perthuis, David Pointcheval



**SNARKs compatible with FHE ciphertexts based on LWE rings**

[EP:2021/322] *Rinocchio: SNARKs for Ring Arithmetic*
Chaya Ganesh, **Anca Nitulescu**, Eduardo Soria-Vazquez

# Outline

SNARKs
Background

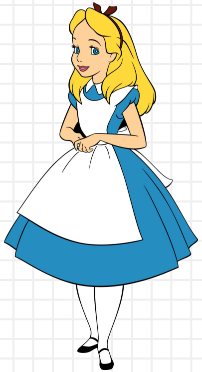
Framework
for Rinocchio

Challenges
New tools

Conclusion

# Introduction to SNARKs

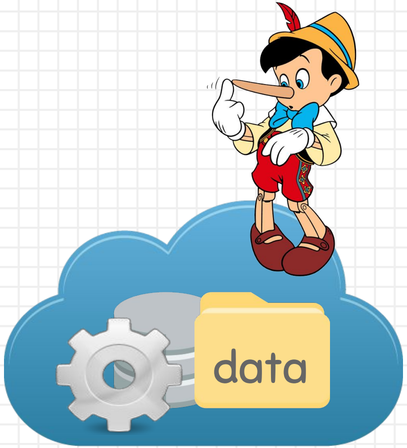
**SNARK** Background | **Framework** for Rinocchio | **Challenges** New tools | **Conclusion**



Verifier

Claim

f(data)= **y**




Prover

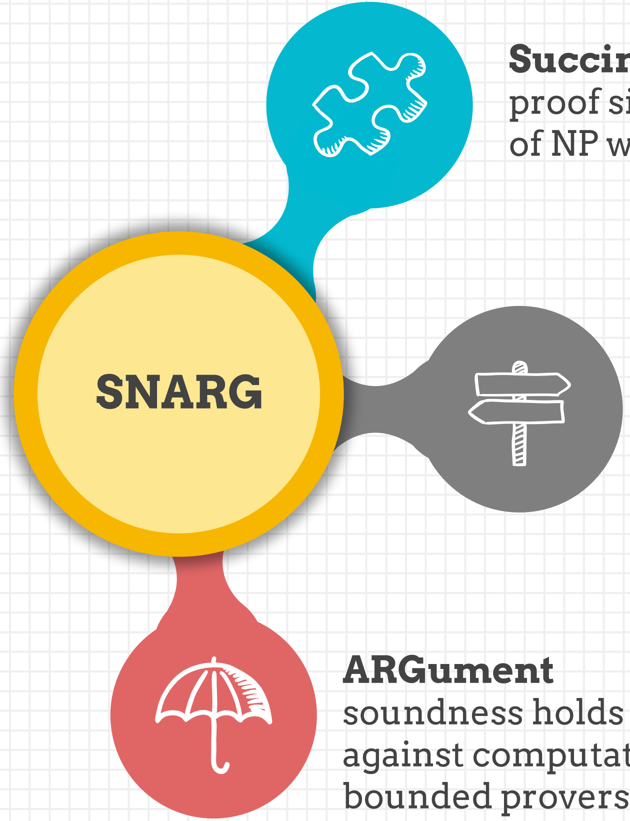
# SNARK: Algorithms


Setup( $\lambda$) $\longrightarrow$ (crs, td)

Prove( crs, f, $\mathbf{x}$, w ) $\longrightarrow$ $\pi$ $\mathcal{R}$ : f($\mathbf{x}$, w) = 1

Verify( crs, f, $\mathbf{x}$, $\pi$ ) $\rightarrow$ 0/1

# **SNARG**: Succinct Non-Interactive ARGument

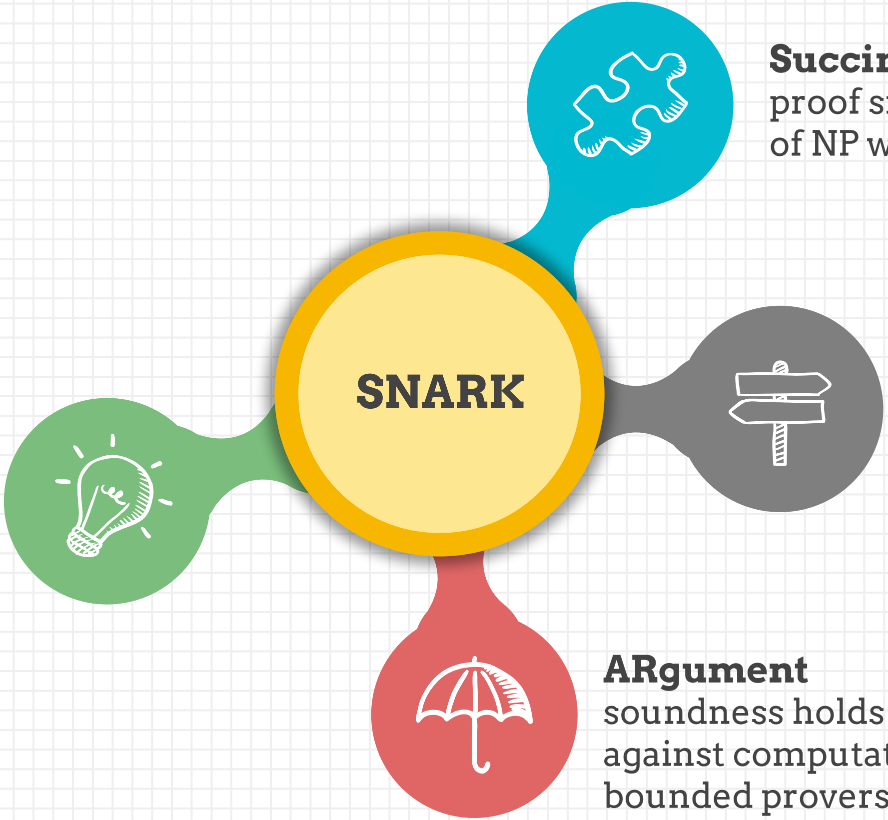



**Succinctness**
proof size independent of NP witness size

**Non-Interactivity**
no exchange between prover and verifier

**ARGument**
soundness holds only against computationally bounded provers

# **SNARK**: Succinct Non-Interactive ARgument of Knowledge




**Succinctness**
proof size independent of NP witness size

**Non-Interactivity**
no exchange between prover and verifier

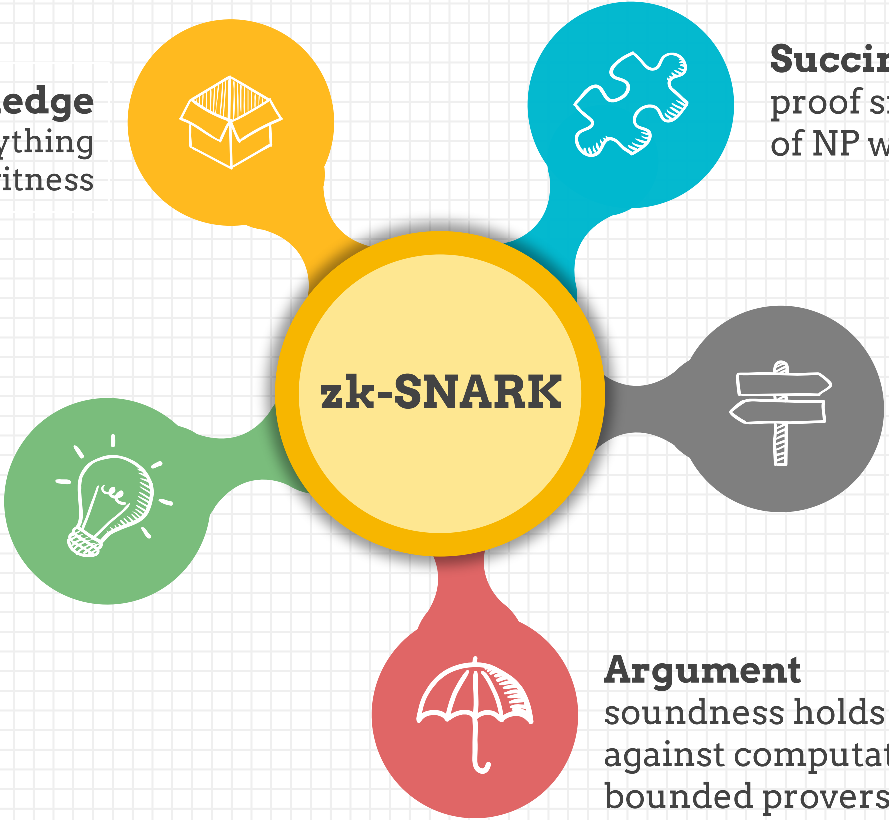
**ARgument**
soundness holds only against computationally bounded provers

**Knowledge Soundness**
a witness can be efficiently extracted from the prover

# **zk-SNARK:** Zero-Knowledge SNARK

**Zero-Knowledge**
does not leak anything about the witness

**Succinctness**
proof size independent of NP witness size




**Non-Interactivity**
no exchange between prover and verifier

**Knowledge Soundness**
a witness can be efficiently extracted from the prover

**Argument**
soundness holds only against computationally bounded provers

# SNARKs: Preprocessing for constant size proofs

R. Gennaro,
C. Gentry, B. Parno, M.
Raykova

**[GGPR13]
QSP and succinct NIZKs
without PCPs**

B. Parno,
J.Howell, C. Gentry,
M. Raykova

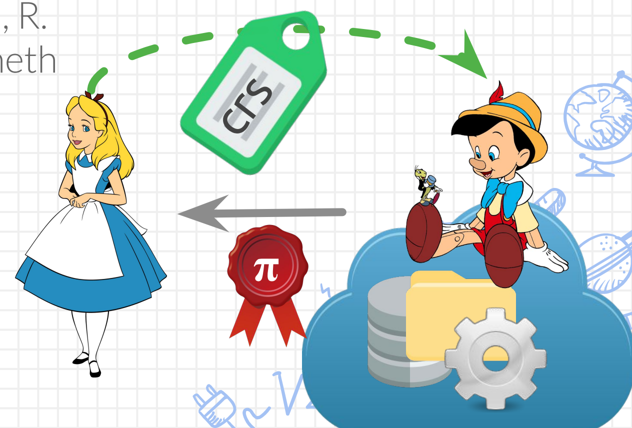
**[PHGR13]
Pinocchio: Nearly practical
verifiable computation**

**[BCI+13] SNARGs via
linear interactive proofs**

N.Bitansky,
A. Chiesa, Y. Ishai, R.
Ostrovsky, O Paneth

**[Groth16] On the Size of
Pairing-based
Non-interactive Arguments**

J. Groth



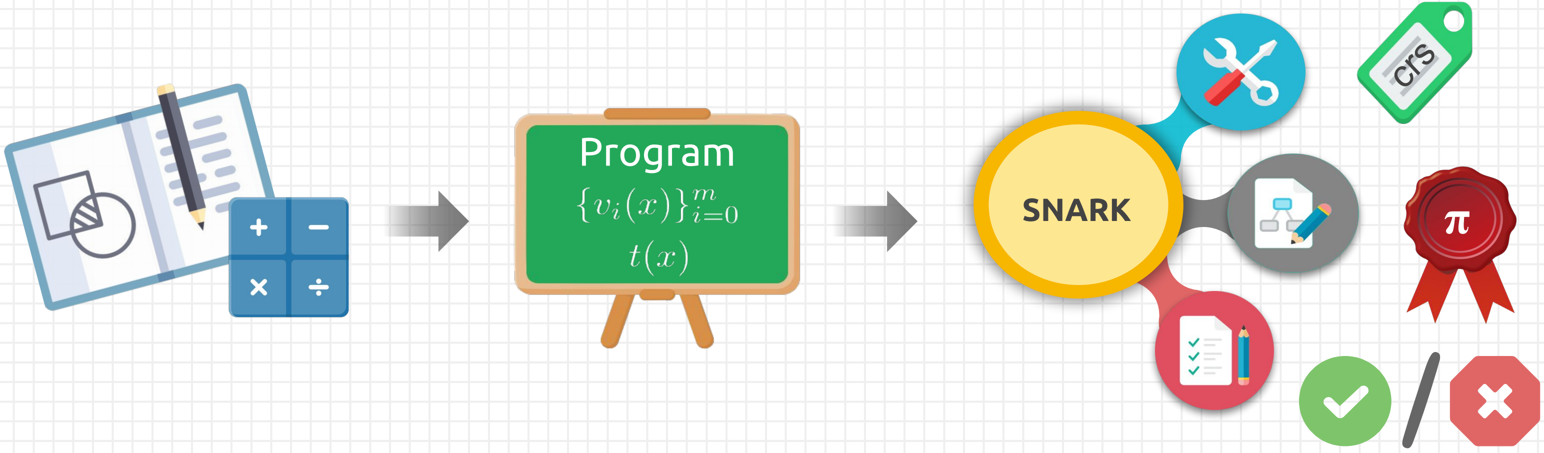
# Key Steps to Build SNARKs

SNARK
Background

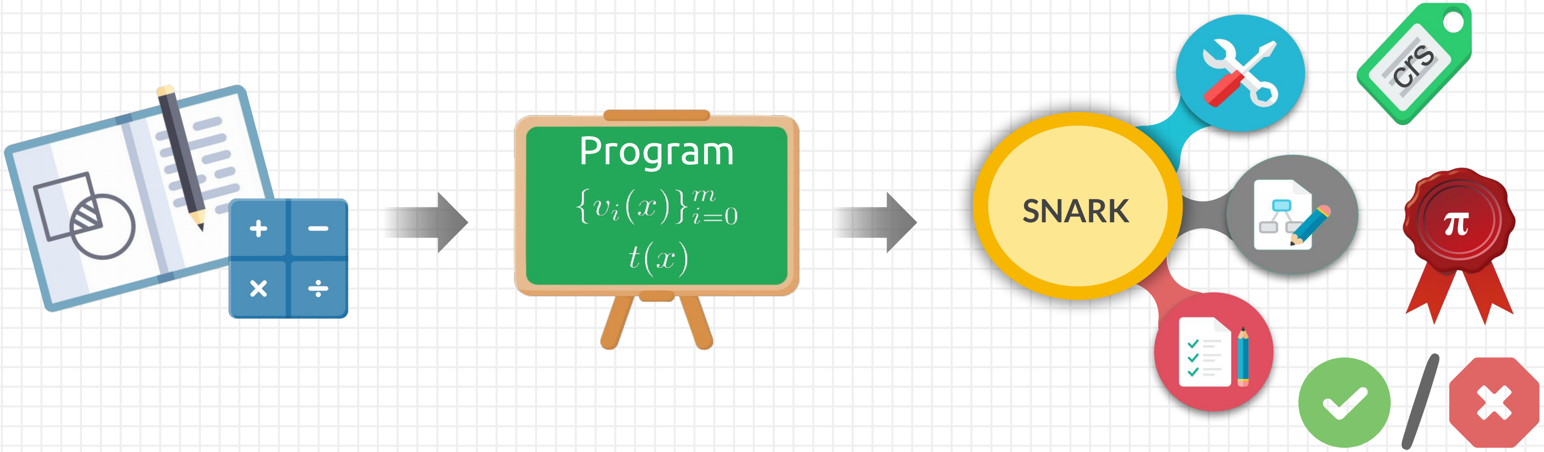
Framework
for Rinocchio

Challenges
New tools

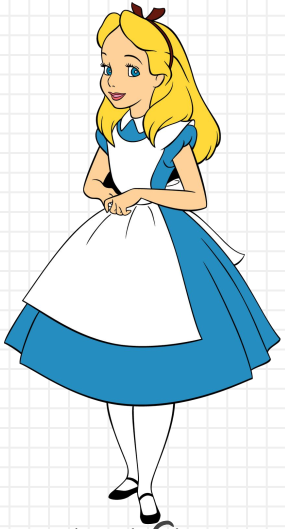
Conclusion



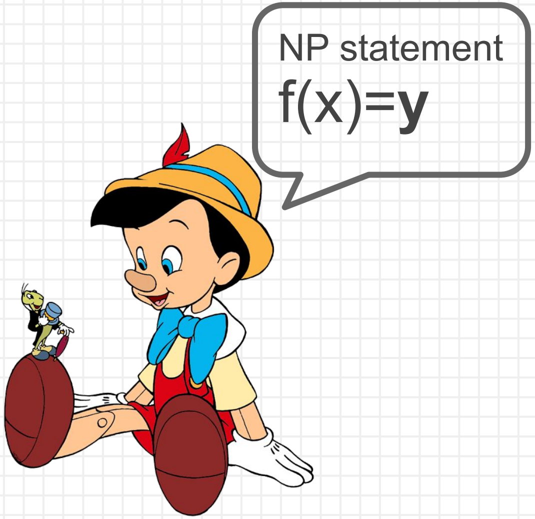
# Frameworks for SNARKs



# Proving NP statements

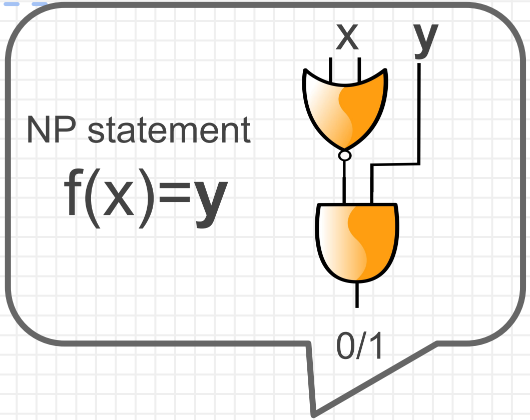


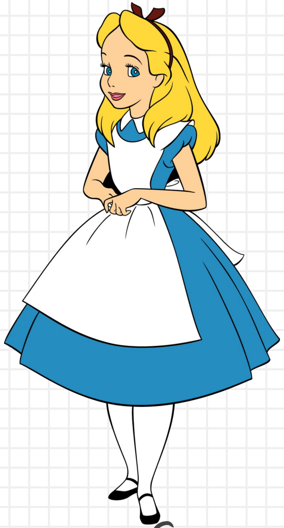
Verifier




Prover

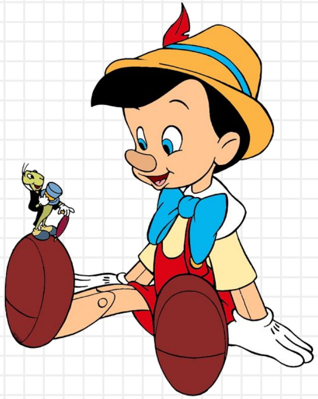
# Computation: Circuit SAT



Verifier

Claim f(x)=y

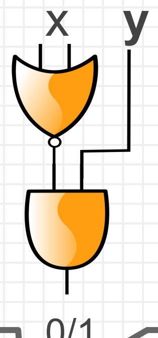


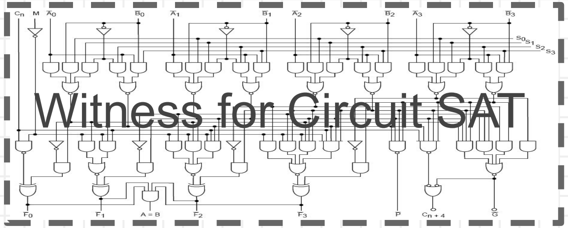
Prover

# NP witness: Too long!

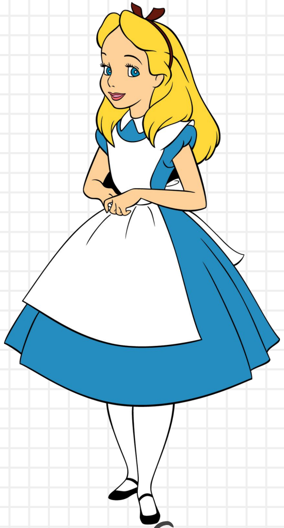
NP statement

$f(x)=\mathbf{y}$

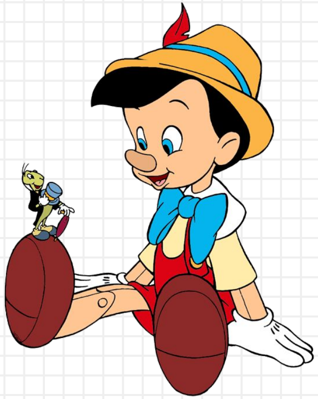



Claim $f(x)=\mathbf{y}$



Verifier

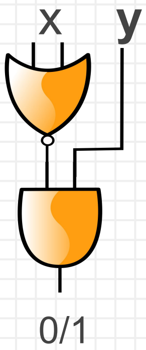


Prover

# Prover solves equivalent problem instead

NP statement
$f(x)=\mathbf{y}$

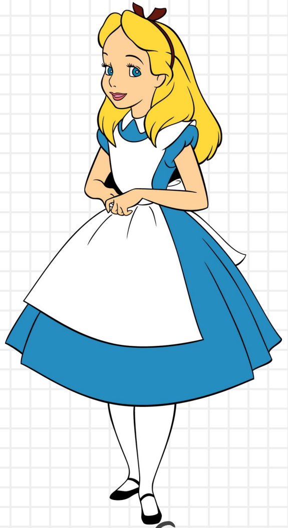
compilation




Circuit SAT solution

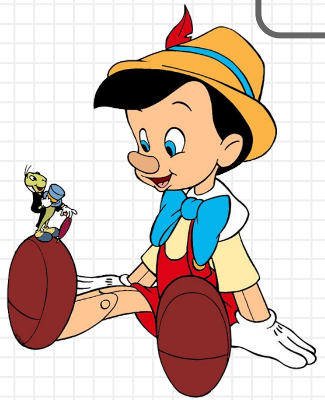
**Polynomial problem**

Given **v(x)**, **t(x)**.
Find **P(x)** such that
$\mathbf{P(x)}t(x) = v(x)$



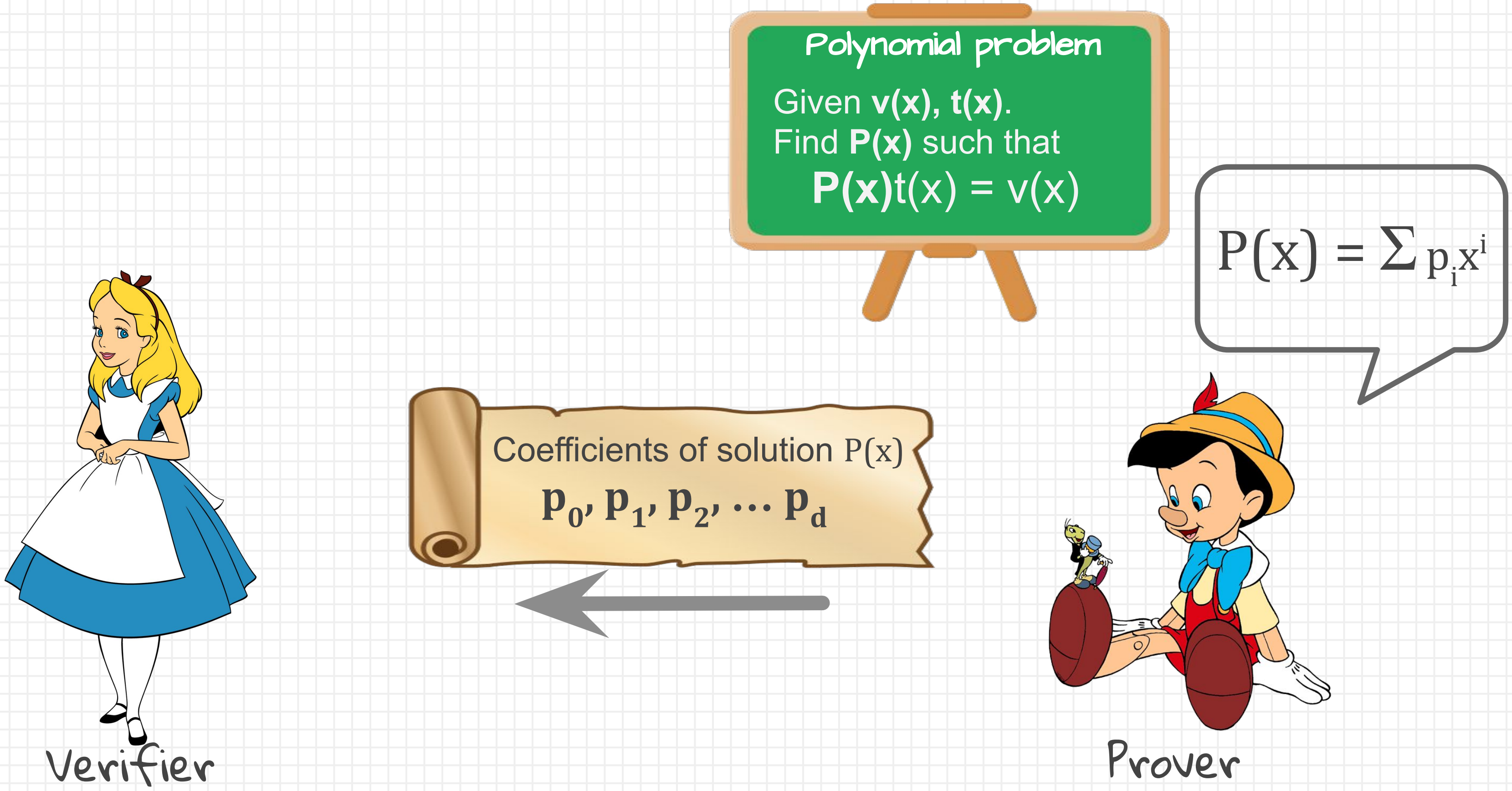
Verifier

P(x)

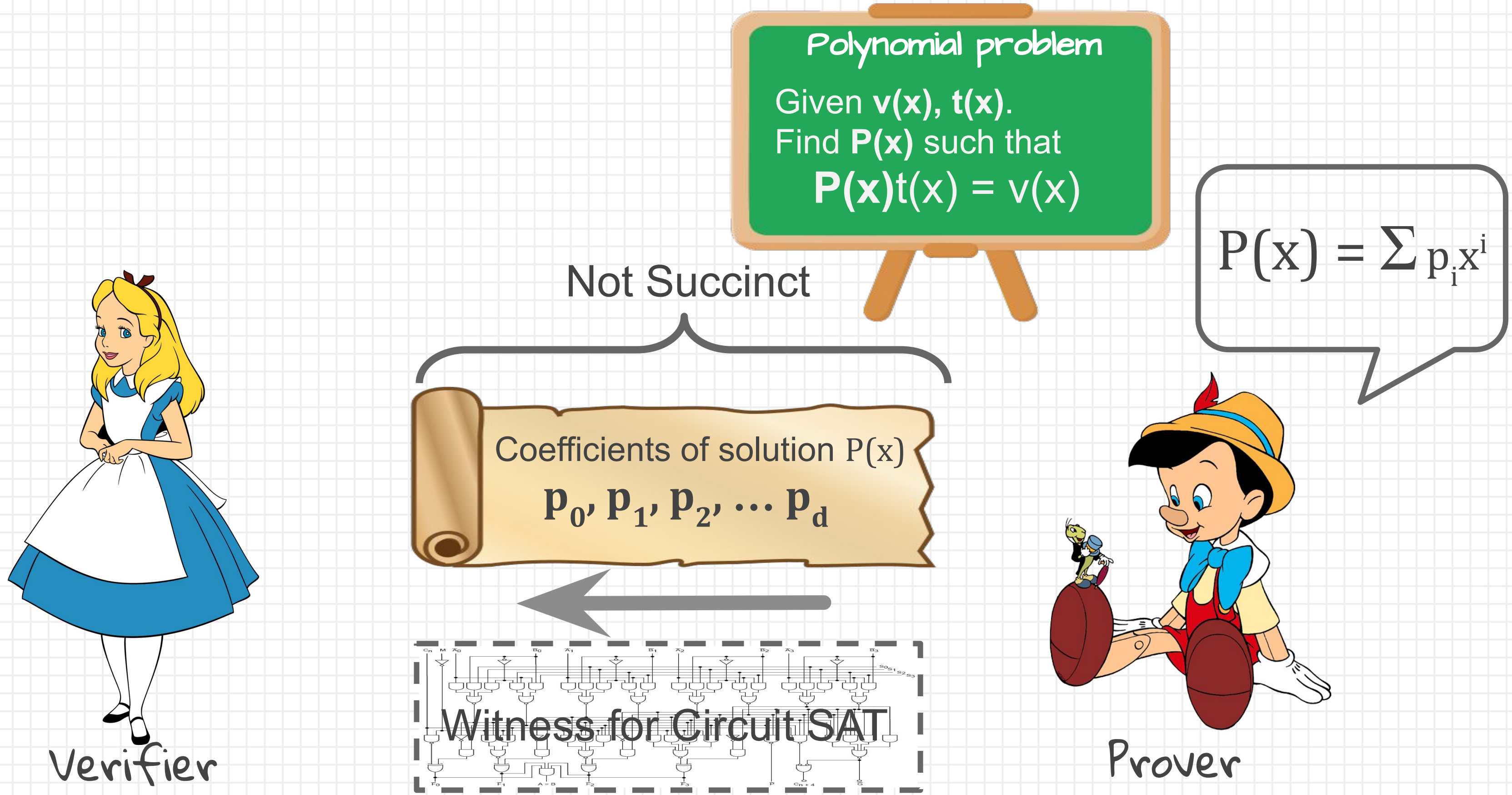


Prover

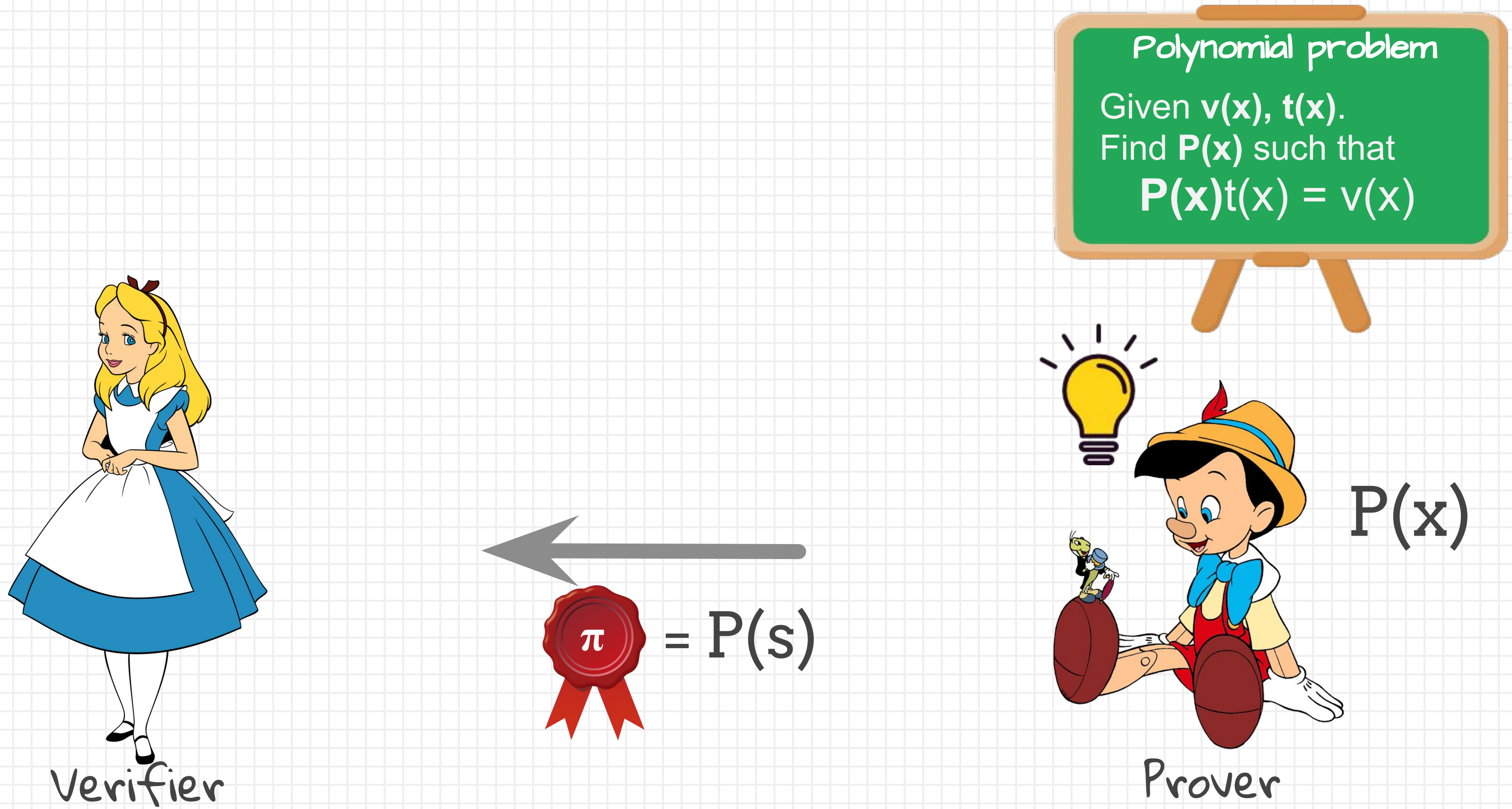
# Prover shows polynomial: too long



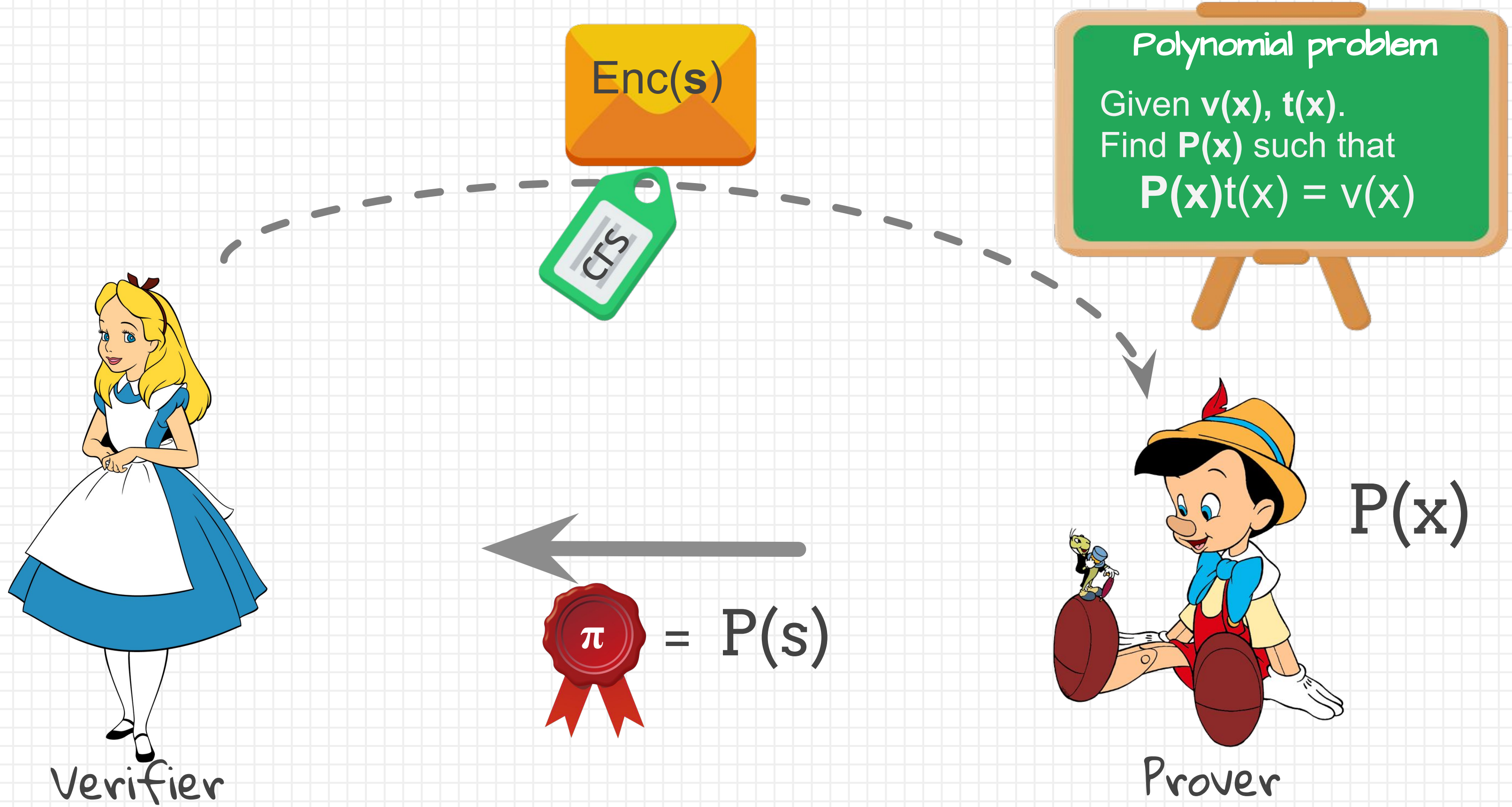
# Prover shows polynomial: too long



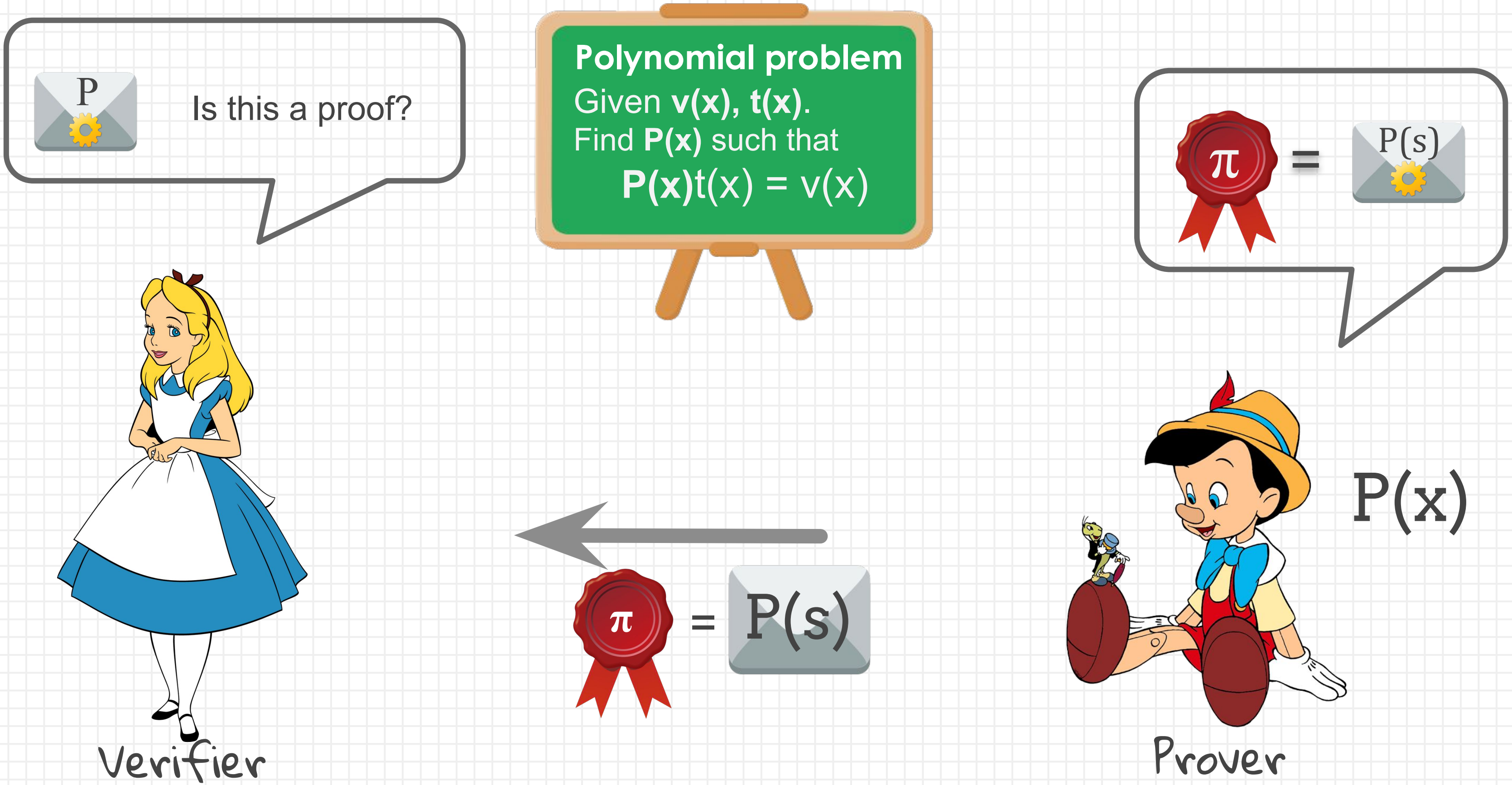
# Evaluate solution at point s



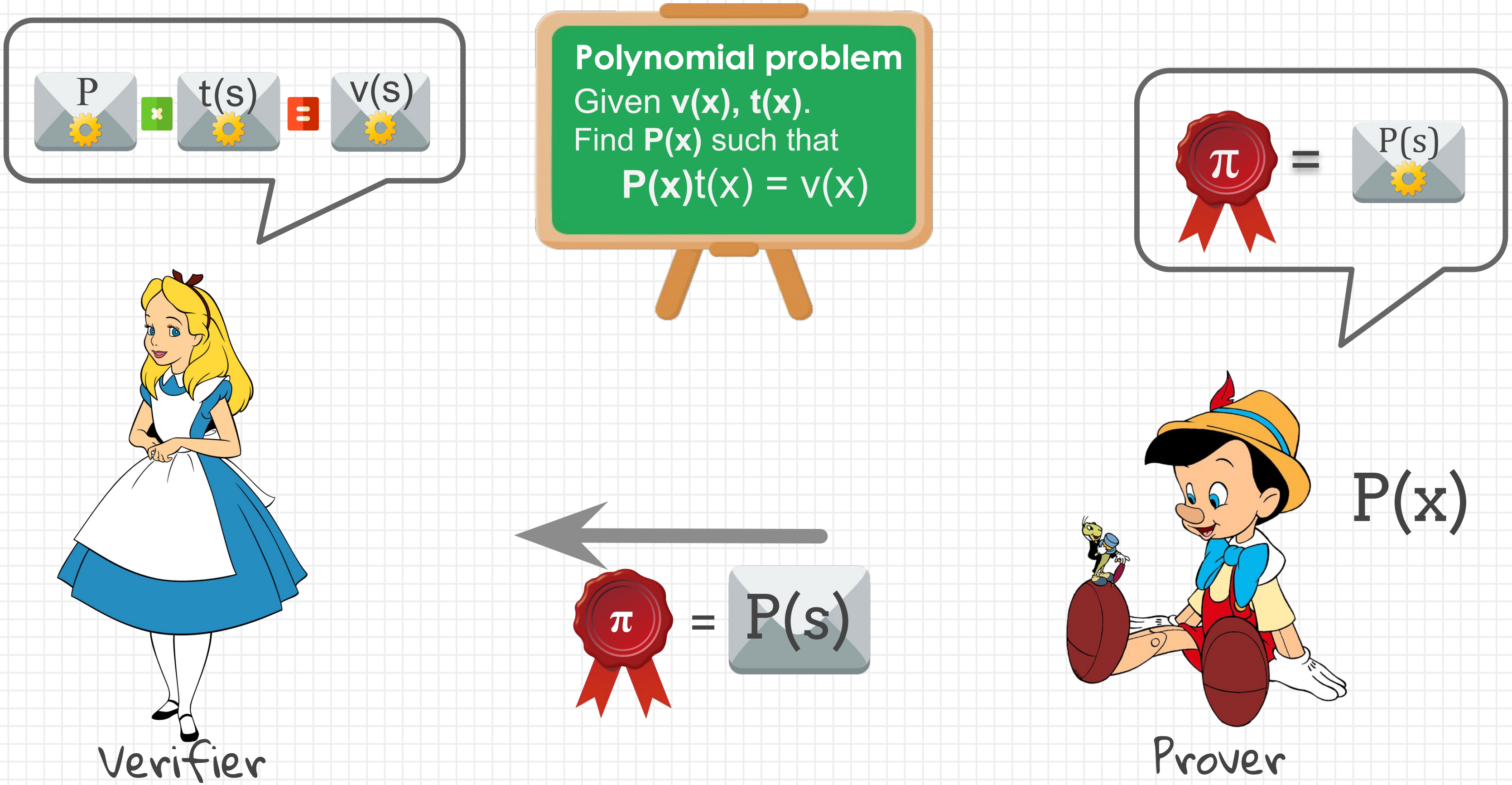
# Evaluate solution at point s



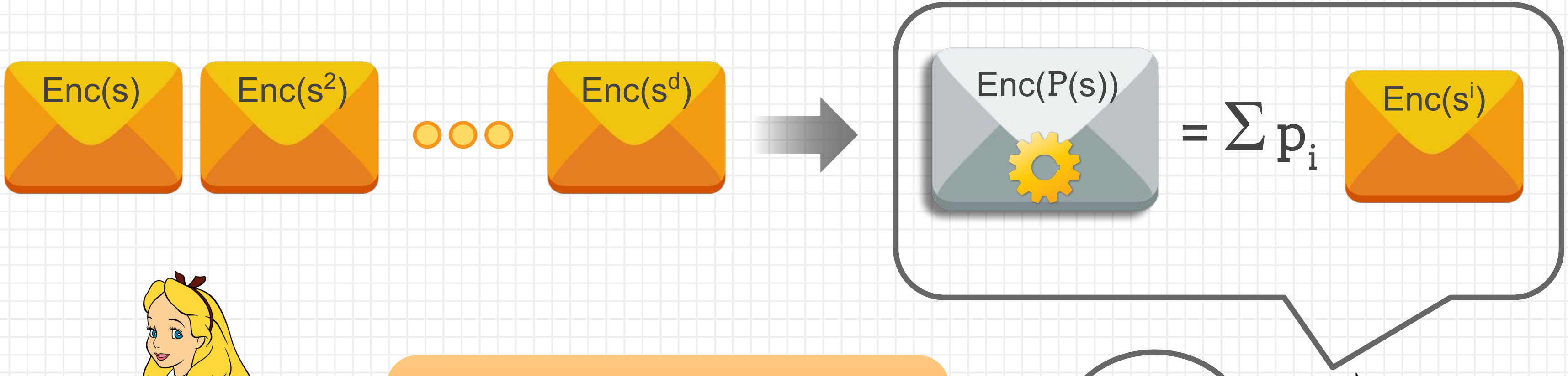
# General SNARK framework

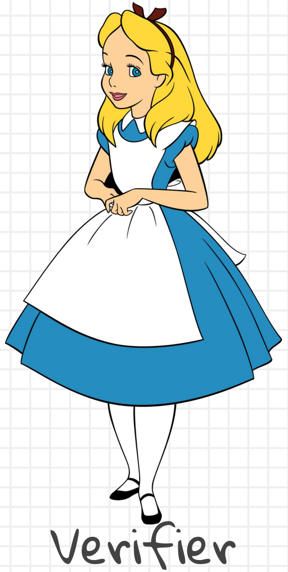


# Verification in a single point



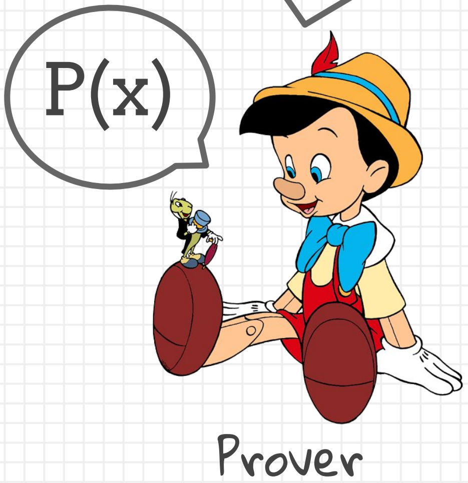
# Encoding Properties for Verification



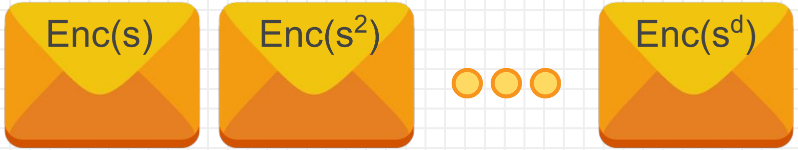



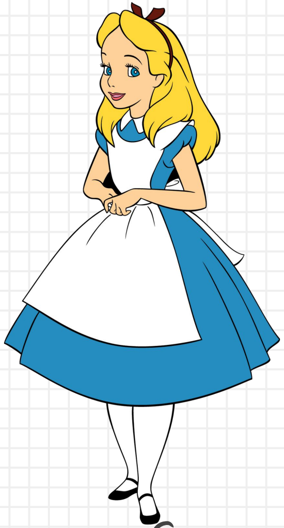

Encoding:

- ***linearly*** homomorphic



# Encoding Properties for Verification

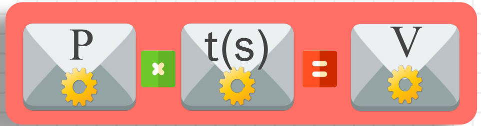


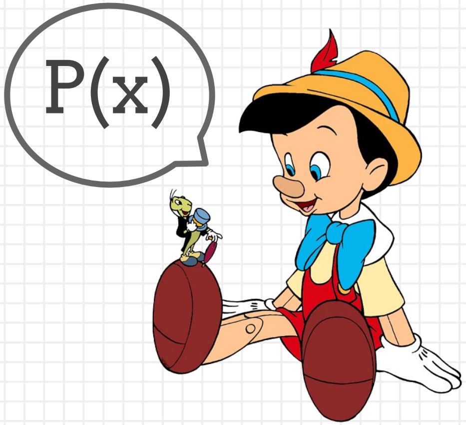



Verifier

Encoding:

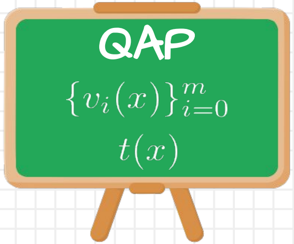
- ***linearly*** homomorphic
- quadratic root detection
- image verification




Prover

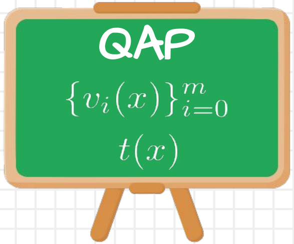
# SNARK: Methodology

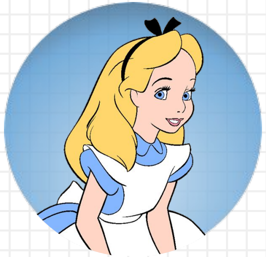




| Target Statement R(y,w)=1 | Computational Model (Representation) | Encodings Secure under Knowledge Assumptions |
|---|---|---|
| Arithmetic Circuit SAT | QAP / SAP over Field = $Z_p$ | **PKE** Power Knowledge of Exponent<br>**GGM** Generic Group Model |

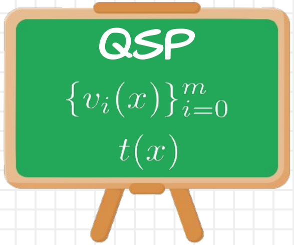
# SNARK: Methodology




| Target Statement R(y,w)=1 | Computational Model (Representation) | **Encodings Secure** under **Knowledge Assumptions** |
|---|---|---|
| Arithmetic Circuit SAT | **QAP / SAP** over **Field = $Z_p$** | **PKE** Power Knowledge of Exponent<br>**GGM** Generic Group Model |
| Boolean Circuit SAT | | |

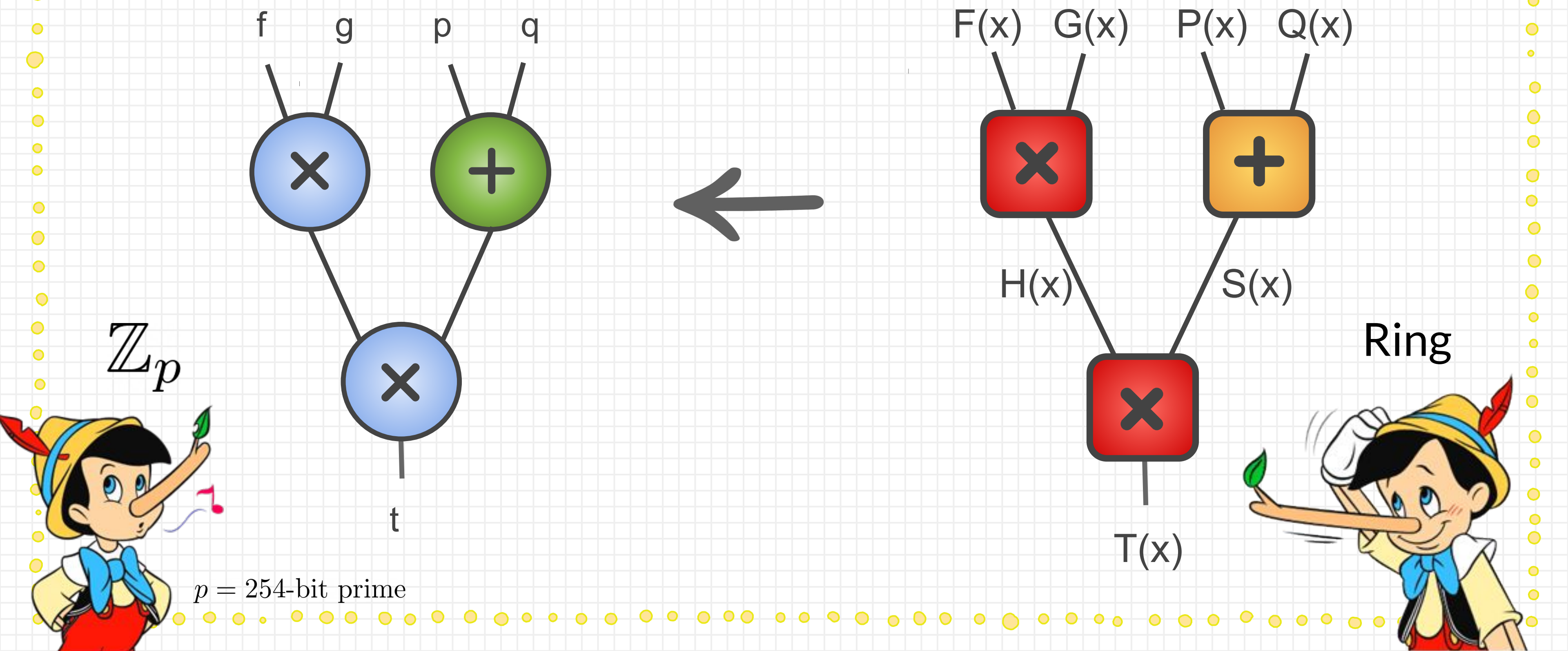
# SNARK: Methodology



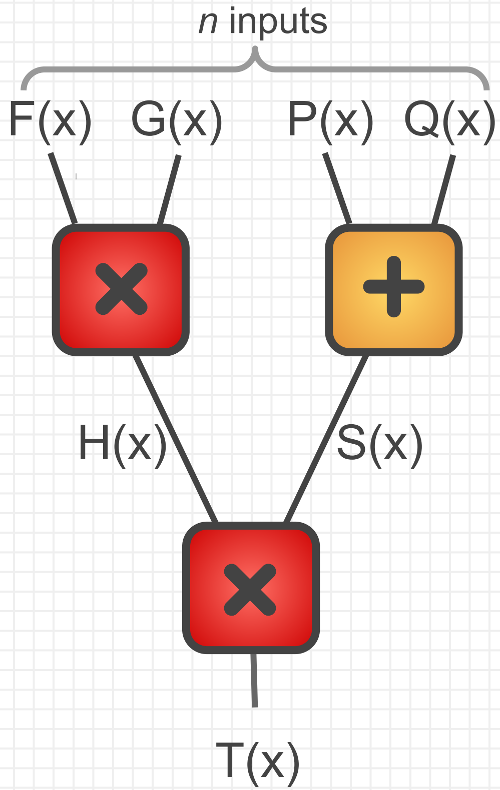


| Target Statement R(y,w)=1 | Computational Model (Representation) | Encodings Secure under Knowledge Assumptions |
|---|---|---|
| Arithmetic Circuit SAT | **QAP / SAP** over **Field = $Z_p$** | **PKE** Power Knowledge of Exponent<br>**GGM** Generic Group Model |
| Boolean Circuit SAT | **QSP / SSP** over **Field = $Z_p$** | **PKE:** Power Knowledge of Exponent |

# Circuit over Field vs Circuit over Ring



# Difficulty: Circuit over FHE ciphertexts




**FHE** [BV11] based on ring-LWE

**Ciphertexts = Polynomials**

$$\mathbb{R}_q = \mathbb{Z}_q[x]/R(x)$$

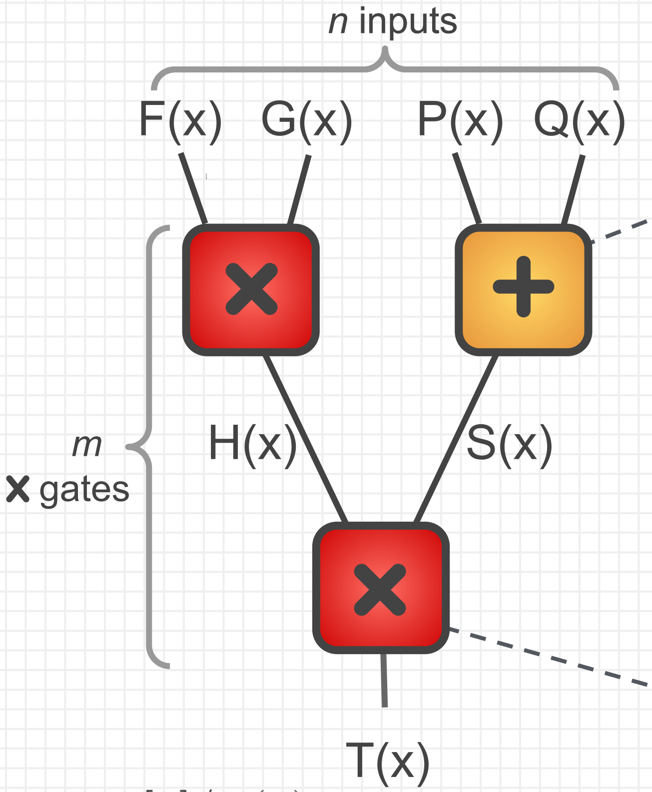
Polynomial additions

&

Polynomial multiplications

*polynomials of degree *d*

# Challenge: Circuit over Polynomials




**$O(n \cdot d)$** scalar additions

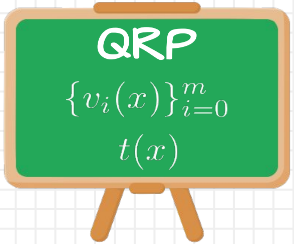
&

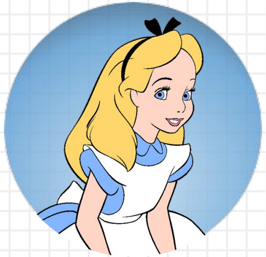
**$O(m \cdot d \cdot \log d)$** scalar multiplications

*for polynomials of degree *d*

$\mathbb{R}_q = \mathbb{Z}_q[x]/R(x)$

# SNARK: Methodology




| Target Statement R(y,w)=1 | Computational Model (Representation) | Encodings Secure under Knowledge Assumptions |
|---|---|---|
| Arithmetic Circuit SAT | **QAP / SAP** over **Field = $Z_p$** | **PKE** Power Knowledge of Exponent<br>**GGM** Generic Group Model |
| Boolean Circuit SAT | **QSP / SSP** over **Field = $Z_p$** | **PKE:** Power Knowledge of Exponent |
| General Circuits over Rings | **QRP over Ring = R** | **Augmented PKE:** Power Knowledge of Encoding |

# Contribution for short

[PHGR13]
Pinocchio: Nearly practical verifiable computation

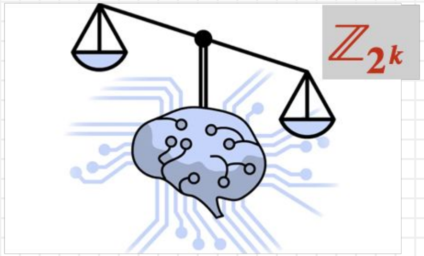
[GNS21]
Rinocchio

# More SNARKs applications



Outsourcing computation
(on encrypted data)


Anonymous Credentials




Fairness in
Machine Learning

# More SNARKs applications

$\mathcal{R}$inocchio

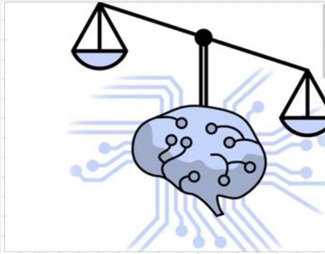
$\mathcal{R}_q$

Outsourcing computation
(on encrypted data)

RSA

$\mathbb{Z}_{p \cdot q}$

Anonymous Credentials



$\mathbb{Z}_{2^k}$

Fairness in
Machine Learning

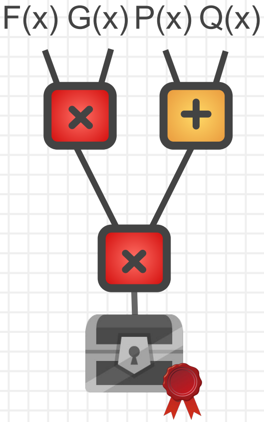
# Technical Details

SNARK
Background

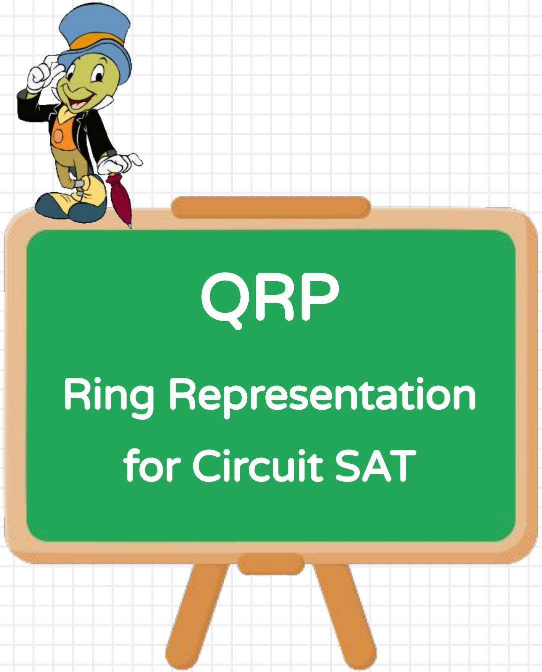
Framework
for Rinocchio

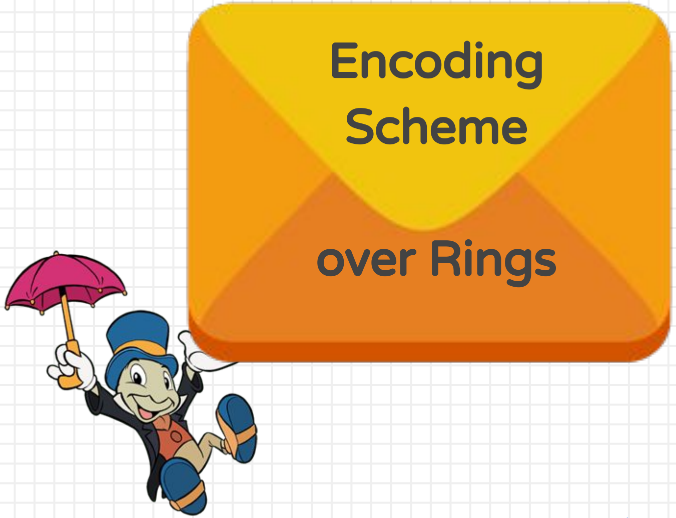
Challenges
New tools

Conclusion




# Main Ingredients for Rinocchio

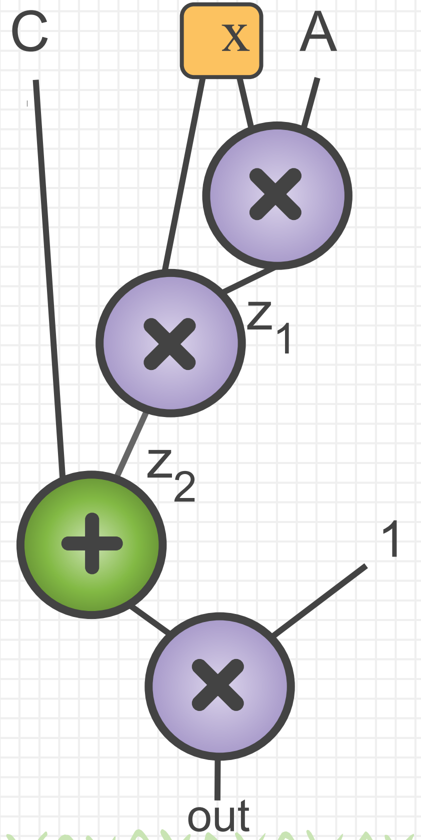


## Example: Solution for equation $Ax^2 + C = 0$

System:

$$\begin{cases} z_1 = A \cdot x \\ z_2 = z_1 \cdot x \\ out = (C + z_2) \cdot 1 \end{cases}$$

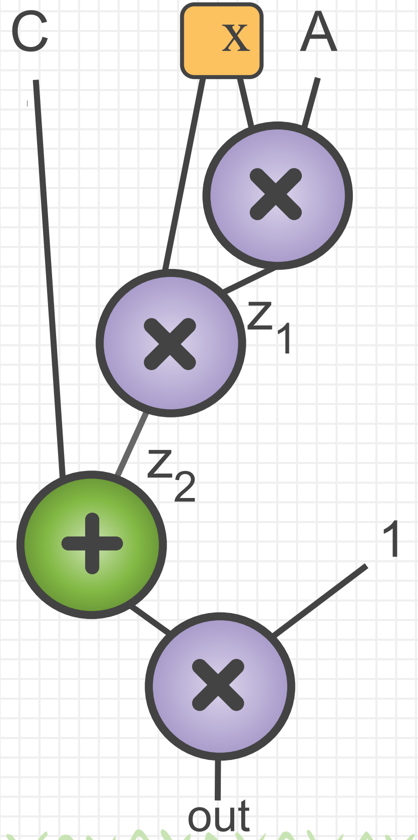


## Example: Solution for equation $Ax^2 + C = 0$

System:

$$\begin{cases} z_1 = A \cdot x \\ z_2 = z_1 \cdot x \\ out = (C + z_2) \cdot 1 \end{cases}$$

$$\mathbf{a} = (1, x, z_1, z_2, out)^T$$



# R1CS for vector $\mathbf{a} = (1, A, C, x, z_1, z_2, out)$

System:

$$\begin{cases} z_1 = A \cdot x \\ z_2 = z_1 \cdot x \\ out = (C + z_2) \cdot 1 \end{cases}$$

$\mathbf{a} = (1, x, z_1, z_2, out)^T$

$\mathbf{a} = (1, x, z_1, z_2, out)^T$

$$(A, 0, 0, 0, 0) \cdot \mathbf{a} \circ (0, 1, 0, 0, 0) \cdot \mathbf{a} = (0, 0, 1, 0, 0) \cdot \mathbf{a}$$

$$(0, 0, 1, 0, 0) \cdot \mathbf{a} \circ (0, 1, 0, 0, 0) \cdot \mathbf{a} = (0, 0, 0, 1, 0) \cdot \mathbf{a}$$

$$(C, 0, 0, 1, 0) \cdot \mathbf{a} \circ (1, 0, 0, 0, 0) \cdot \mathbf{a} = (0, 0, 0, 0, 1) \cdot \mathbf{a}$$

$$(V \mathbf{a}) \circ (W \mathbf{a}) = Y \mathbf{a}$$

## R1CS for vector **a** =(1,A, C, x, $z_1$, $z_2$, out)

System:

$$\begin{cases} z_1 = A \cdot x \\ z_2 = z_1 \cdot x \\ out = (C + z_2) \cdot 1 \end{cases}$$

**a** =(1, x, $z_1$, $z_2$, out)$^T$


$$v_i(r_j) = V_{ji}$$

$$\forall \{r_j\} \in \mathbb{F}^d$$

$$\left(\sum_{i=0}^{m} a_i v_i(x)\right)\left(\sum_{i=0}^{m} a_i w_i(x)\right) = \left(\sum_{i=0}^{m} a_i y_i(x)\right)$$

# Proving a Solution for Equation **$Ax^2 + C = 0$**

System:
$$\begin{cases} z_1 = A \cdot x \\ z_2 = z_1 \cdot x \\ out = (C + z_2) \cdot 1 \end{cases}$$

$$\mathbf{a} = (1, x, z_1, z_2, out)^T$$


$$v_i(r_j) = V_{ji}$$

$$\forall \{r_j\} \in \mathbb{F}^d$$

$$\left(\sum_{i=0}^{m} a_i v_i(x)\right)\left(\sum_{i=0}^{m} a_i w_i(x)\right) = \left(\sum_{i=0}^{m} a_i y_i(x)\right)$$

$$\prod_{j=1}^{d}(x - r_j) \,\Big|\, \left(\sum_{i=0}^{m} a_i v_i(x)\right)\left(\sum_{i=0}^{m} a_i w_i(x)\right) - \left(\sum_{i=0}^{m} a_i y_i(x)\right)$$

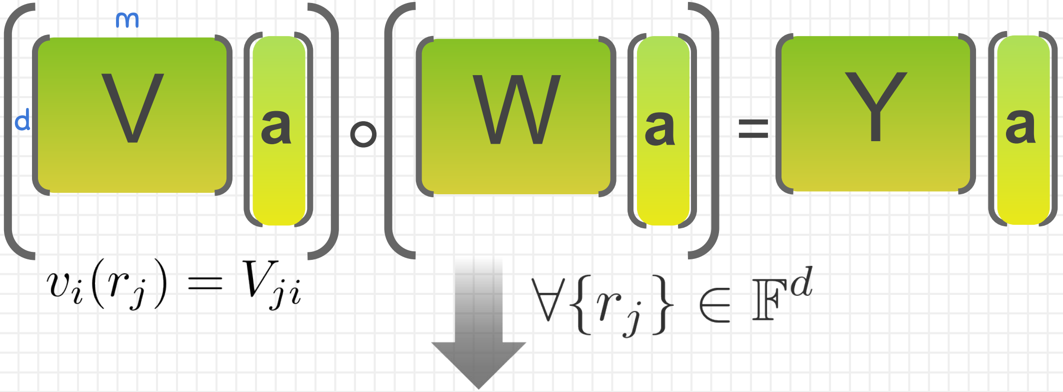
# Proving a Solution for Equation **Ax² + C = 0**

Given $\{v_i(x)\}_i, \{w_i(x)\}_i,$
$\{y_i(x)\}_i, t(x)$

Find $V(x), W(x), Y(x), h(x)$
s.t.

$$V(x) = \sum_{i=0}^{m} a_i v_i(x)$$

$$t(x)h(x) = V(x)W(x) - Y(x)$$




$$v_i(r_j) = V_{ji}$$

$$\forall \{r_j\} \in \mathbb{F}^d$$

$$\Big(\sum_{i=0}^{m} a_i v_i(x)\Big)\Big(\sum_{i=0}^{m} a_i w_i(x)\Big) = \Big(\sum_{i=0}^{m} a_i y_i(x)\Big)$$

$$\prod_{j=1}^{d}(x - r_j) \,\Big|\, \Big(\sum_{i=0}^{m} a_i v_i(x)\Big)\Big(\sum_{i=0}^{m} a_i w_i(x)\Big) - \Big(\sum_{i=0}^{m} a_i y_i(x)\Big)$$

# Polynomial Equation with Coefficients in a Ring

$$t(x) = \prod_{j=1}^{d}(x - r_j) \;\Big|\; \Big(\sum_{i=0}^{m} a_i v_i(x)\Big)\Big(\sum_{i=0}^{m} a_i w_i(x)\Big) - \Big(\sum_{i=0}^{m} a_i y_i(x)\Big) = p(x)$$

Necessary property over Rings for **Ideals** $I_j = (x - r_j)$

Isomorphism for **QRP** soundness ⇔ **Ideals** $I_j$ are co-prime:

$$\frac{R[x]}{(t(x))} \simeq \frac{R[x]}{I_1} \times \ldots \times \frac{R[x]}{I_d} \simeq R \times \ldots \times R$$

$$p(x) \longmapsto (p_1(x), \ldots, p_d(x)) \longmapsto (p(r_1), \ldots, p(r_d))$$

# Polynomial Equation with Coefficients in a Ring

Works for $\mathrm{R} = \mathbb{F}$, as then the ideals $I_j$ are co-prime.

$$\ldots - r_j) \;\Big|\; \Big(\sum_{i=0}^{m} a_i v_i(x)\Big)\Big(\sum_{i=0}^{m} a_i w_i(x)\Big) - \Big(\sum_{i=0}^{m} a_i y_i(x)\Big)$$

Necessary p… erty over Rings for **Ideals** $I_j = (x - r_j)$

Isomorphism fo… RP soundness ⇔ **Ideals** $I_j$ are co-prime:

$$\frac{R[x]}{(t(x))} \simeq \frac{R[x]}{I_1} \times \ldots \times \frac{R[x]}{I_d} \simeq R \times \ldots \times R$$

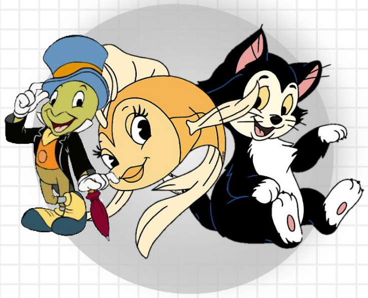
$$p(x) \longmapsto (p_1(x), \ldots, p_d(x)) \longmapsto (p(r_1), \ldots, p(r_d))$$

# Exceptional Sets: to the rescue!

**Def:** Let $R$ be a commutative ring. A set $\mathbf{A} = \{g_1, \ldots, g_n\} \subset R$ is ***exceptional*** iff:

$$\forall\ i \neq j,\ (g_i - g_j) \in R^*$$

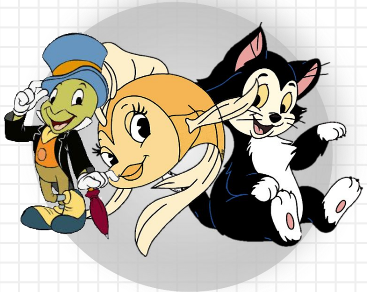
Exceptional sets have **no further** algebraic **structure.**
Not even closure!



# Exceptional Sets

**Def:** Let $R$ be a commutative ring. A set $\mathbf{A} = \{g_1, \ldots, g_n\} \subset R$ is ***exceptional*** iff:

$$\forall\, i \neq j,\ (g_i - g_j) \in R^*$$



Exceptional sets have **no further** algebraic **structure.** Not even closure!

Given exceptional set $\mathbf{A}$, the **ideals** $I_j = \left(x - g_j\right)$ are **pairwise co-prime** (i.e. $\forall i \neq j,\ \ I_i + I_j = R[X]$).

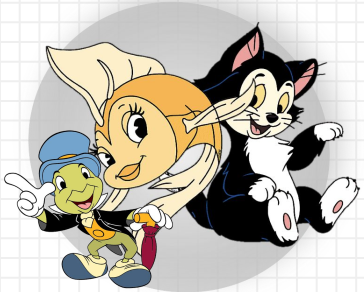
- Proof: $-\left(x - g_i\right) + (x - g_j) = (g_i - g_j) \in R^*$
- Meaning: We can apply CRT in R[X], for big enough $\mathbf{A} \subset R$.

# Exceptional Sets

**Def:** Let $R$ be a commutative ring. A set $\mathbf{A} = \{g_1, \dots, g_n\} \subset R$ is ***exceptional*** iff:

$$\forall\, i \neq j,\ (g_i - g_j) \in R^*$$

Exceptional sets have **no further** algebraic **structure.** Not even <u>closure</u>!



Given exceptional set $\mathbf{A}$, the **ideals** $I_j = \left(x - g_j\right)$ are **pairwise co-prime** (i.e. $\forall i \neq j,\ \ I_i + I_j = R[X]$).

$$\frac{R[x]}{(t(x))} \simeq \frac{R[x]}{I_1} \times \dots \times \frac{R[x]}{I_d} \simeq R \times \dots \times R$$

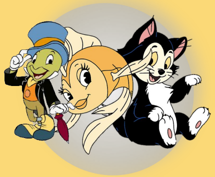
$$\mathrm{p}(x) \longmapsto \left(p_1(x),\ \dots, p_d(x)\right) \longmapsto \left(p\left(g_1\right),\ \dots,\ p(g_d)\right)$$

# Schwartz-Zippel Lemma over Rings

$$t(x) = \prod_{j=1}^{d}(x - r_j) \;\Bigg|\; \Big(\sum_{i=0}^{m} a_i v_i(x)\Big)\Big(\sum_{i=0}^{m} a_i w_i(x)\Big) - \Big(\sum_{i=0}^{m} a_i y_i(x)\Big) = p(x)$$

**Lemma:** Let f $\in R[X]$ be a **non-zero** poly.

$$\Pr_{s \leftarrow A}[f(s) = 0] \leq \frac{\deg(f)}{|A|}$$

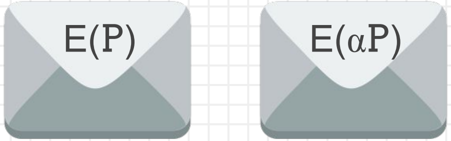


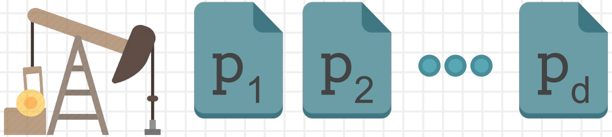
# Encodings

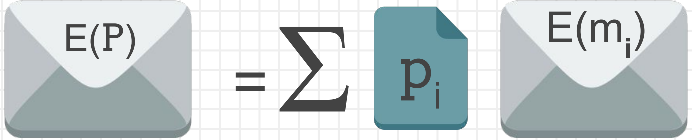
Properties

Assumptions

# Extractable Linear-Only [BISW17]

# Encodings over Fields

DLog Group $\mathbb{G}$

$$\langle g \rangle = \mathbb{G}, \;\; Enc(s) = g^s$$


$$Enc(p(s)) = g^{p(s)}$$

$$g^{\sum_i p_i s^i} = \prod (g^{s^i})^{p_i}$$

# DLog vs General Encoding

DLog Group $\mathbb{G}$

$\langle g \rangle = \mathbb{G}, \; Enc(s) = g^s$

$g^s$ $g^{s^2}$ … $g^{s^d}$

$$Enc(p(s)) = g^{p(s)}$$

$$g^{\sum_i p_i s^i} = \prod (g^{s^i})^{p_i}$$

Encode: $E_{pk}(m) = c$

Decode: $D_{sk}(c) = m$

$E_{pk}(s)$ $E_{pk}(s^2)$ … $E_{pk}(s^d)$

$$Enc(p(s)) = E_{pk}(p(s))$$

$$E_{pk}(\sum p_i s^i) = \sum p_i E_{pk}(s^i)$$

# Quadratic Root Detection – Pairings

$$\langle g \rangle = \mathbb{G}, \langle \tilde{g} \rangle = \tilde{\mathbb{G}}$$

$$Enc(s) = g^s \qquad e : \mathbb{G} \times \mathbb{G} \rightarrow \tilde{\mathbb{G}}$$

$$e(g^a, g^b) = \tilde{g}^{ab}$$

Quadratic root detection **public**

$$t(s)h(s) \stackrel{?}{=} p(s)$$

$$e(g^{t(s)}, g^{h(s)}) \stackrel{?}{=} e(g^{p(s)}, g)$$

# Publicly Verifiable vs Designated Verifiable

$$\langle g \rangle = \mathbb{G}, \langle \tilde{g} \rangle = \tilde{\mathbb{G}}$$

$$Enc(s) = g^s \qquad e : \mathbb{G} \times \mathbb{G} \rightarrow \tilde{\mathbb{G}}$$

$$e(g^a, g^b) = \tilde{g}^{ab}$$

Quadratic root detection **public**

$$t(s)h(s) \stackrel{?}{=} p(s)$$

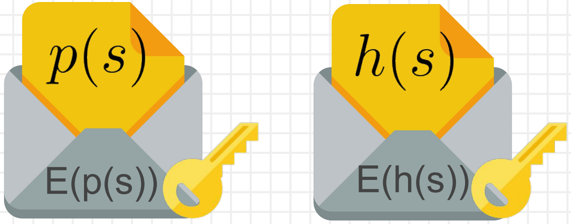
$$e(g^{t(s)}, g^{h(s)}) \stackrel{?}{=} e(g^{p(s)}, g)$$

Encode: $E_{pk}(m) = c$

Decode: $D_{sk}(c) = m$

Quadratic root detection needs **sk**

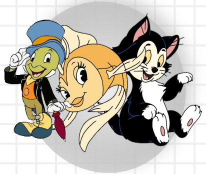
$$t(s)h(s) \stackrel{?}{=} p(s)$$



# Encoding Instantiation for LWE Rings

Rings of the form $\mathcal{R}_q = \mathbb{Z}_q[X]/(h(X))$.

TFHE $\mathbb{R}_q \approx \mathbb{Z}_q^n \quad \mathbb{Z}_q \approx q^{-1}\mathbb{Z}/\mathbb{Z}$



$= \{0, \ldots, p_1 - 1\} \subset \mathcal{R}_q; \quad q = \prod p_i$ s.t. $p_1 < p_2 < \ldots$

**Advantages & Future directions:**

- ✘ Supports "somewhat homomorphic" variants of **BGV** [BGV12] and **FV** [FV12]
- ✘ Allows for significantly better choices for RLWE parameters
- ✘ First SNARK to support rings with q ≠ prime → more expressive FHE
- ✘ We enable new FHE operations → new circuits for plaintext packing, modulo switching
- ✘ **:(** We are only designated-verifier, we don't support Bootstrapping operations

# Encoding Instantiation for LWE Rings

Rings of the form $\mathcal{R}_q = \mathbb{Z}_q[X]/(h(X))$.

$q = \prod p_i$ s.t. $p_1 < p_2 < \dots$

TFHE $\mathbb{R}_q \approx \mathbb{Z}_q^n$ $\mathbb{Z}_q \approx q^{-1}\mathbb{Z}/\mathbb{Z}$

Performance Evaluation:

- ✘ # levels **L** for **BGV** [BGV12] and **FV** [FV12]
- ✘ TFHE performance for such plaintexts?

Possible parameters:

| Scheme | L | n | $\lvert p_1\rvert$ | $\lvert q\rvert$ |
|---|---|---|---|---|
| BGV | 2 | $2^{12}$ | 47 | 109 |
| FV | 2 | $2^{13}$ | 48 | 218 |
| BGV | 4 | $2^{13}$ | 48 | 218 |
| FV | 4 | $2^{14}$ | 51 | 438 |
| BGV | 6 | $2^{14}$ | 51 | 438 |
| FV | 6 | $2^{14}$ | 51 | 438 |
| BGV | 8 | $2^{15}$ | 51 | 881 |
| FV | 8 | $2^{14}$ | 50 | 438 |
| BGV | 10 | $2^{15}$ | 56 | 881 |
| FV | 10 | $2^{15}$ | 56 | 881 |
| BGV | 12 | $2^{15}$ | 59 | 881 |
| FV | 12 | $2^{15}$ | 57 | 881 |

# [PKC:FNP20] SNARK approach

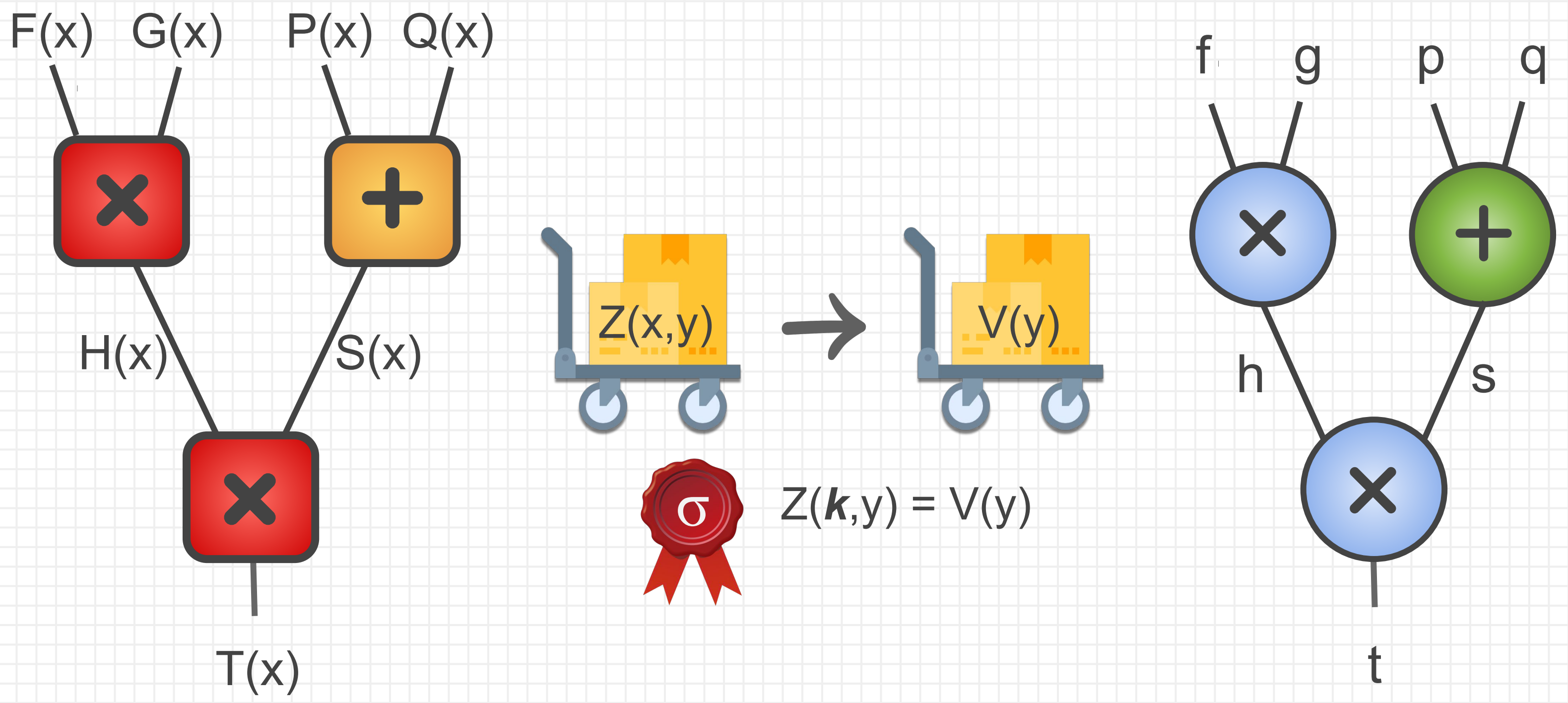

**Compactly Commit to Polynomials**

**ZK Proof** for evaluation in random point **k**

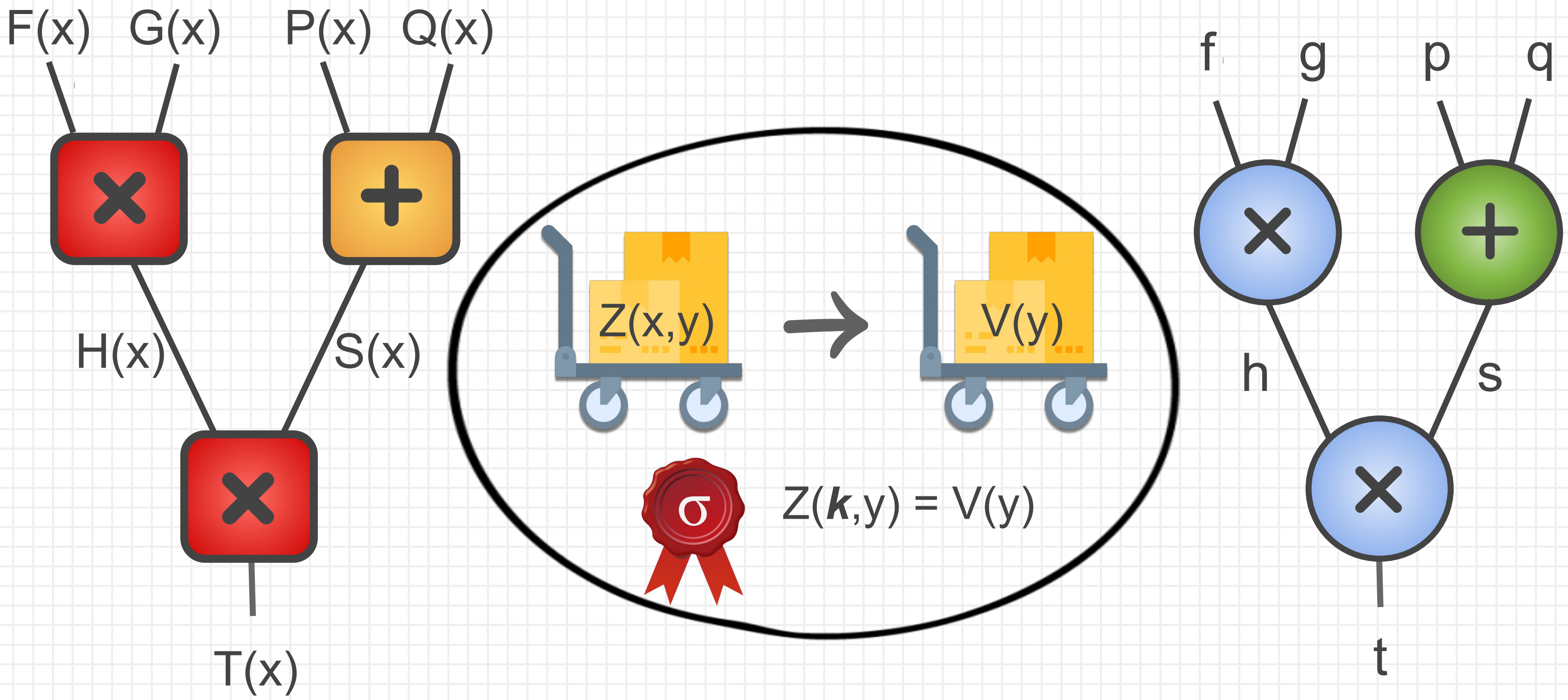
**CaP zk-SNARK**
for arithmetic circuit over scalars

Verifiable Computation with **Privacy**

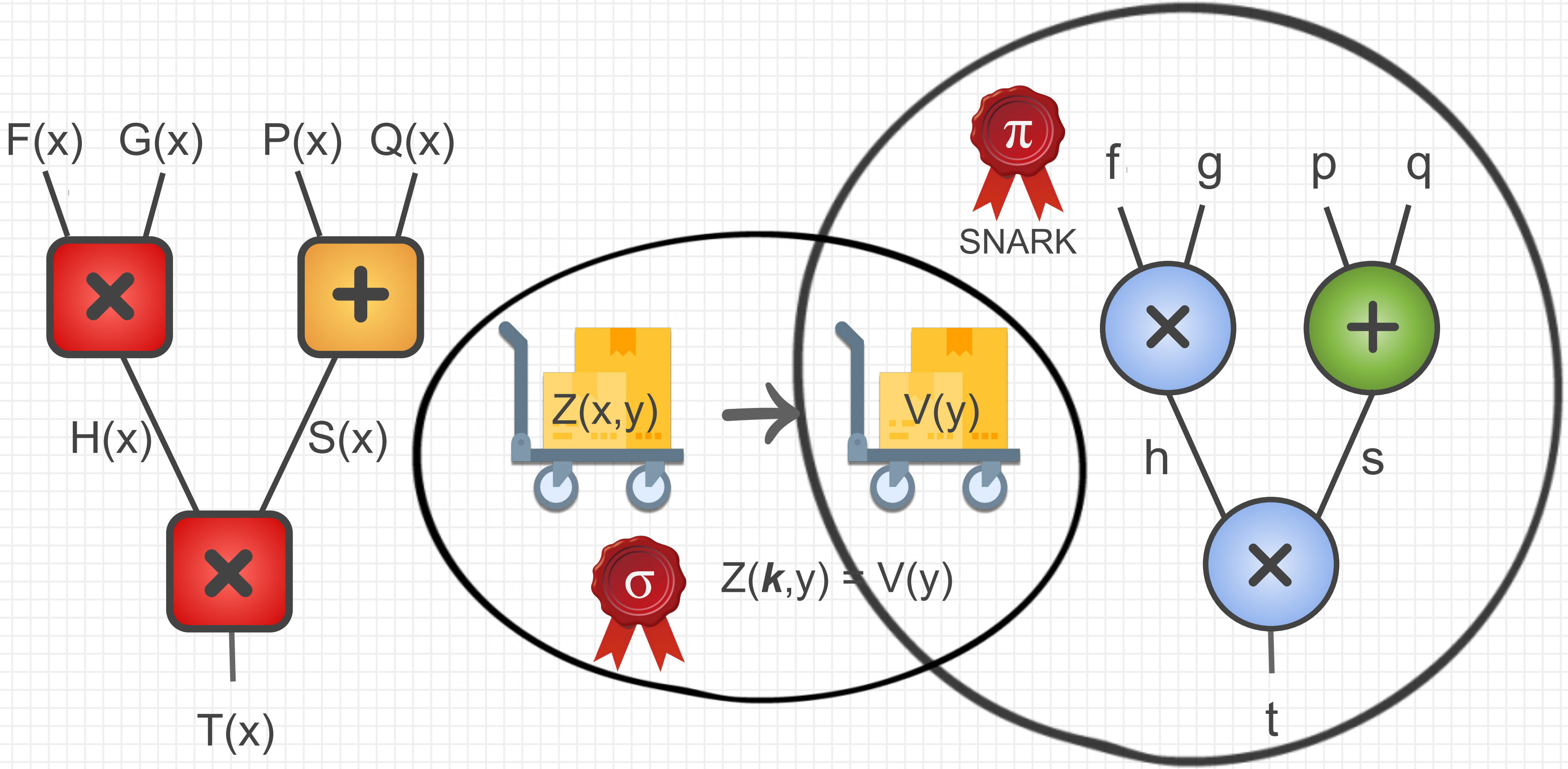
# Proof of Many Evaluations



# Proof of Many Evaluations



# Proof of Arithmetic Circuit over Scalars



# FHE Arithmetics: tailored SNARKs

## [FVP20] Boosting Verifiable Computation on Encrypted Data

Dario Fiore, Anca Nitulescu, David Pointcheval

- ✘ Only supports rings of polynomials $\mathbb{R}_q = \mathbb{Z}_q[x]/R(x)$ for q prime → inefficient FHE
- ✘ Does not support operations for bootstrapping, rescaling etc. in FHE

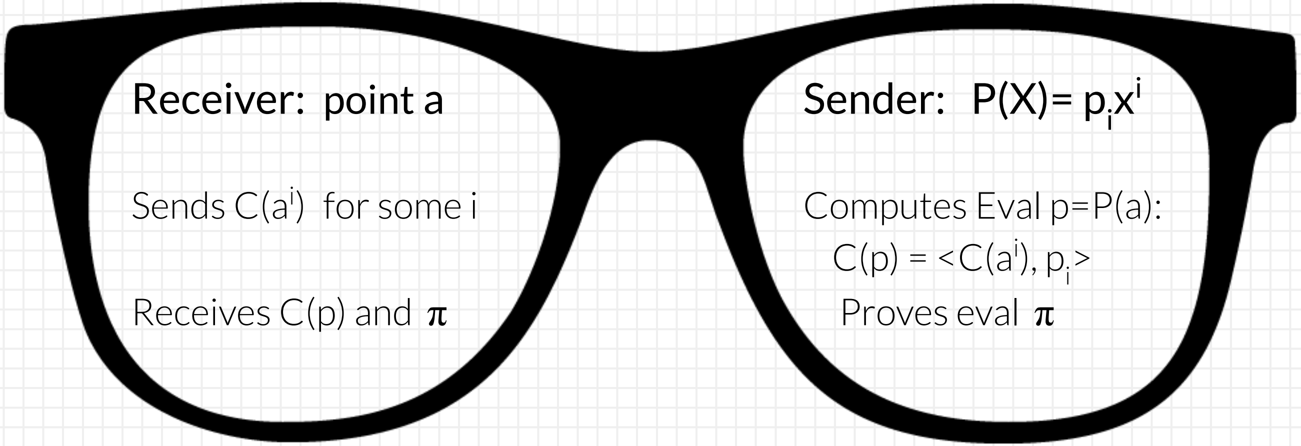
- ✘ Modular - Commit&Proof Composition
- ✘ Publicly Verifiable, anyone can verify without key
- ✘ Zero-Knowledge for inputs and computation

# More specific FHE computations: MyOPE

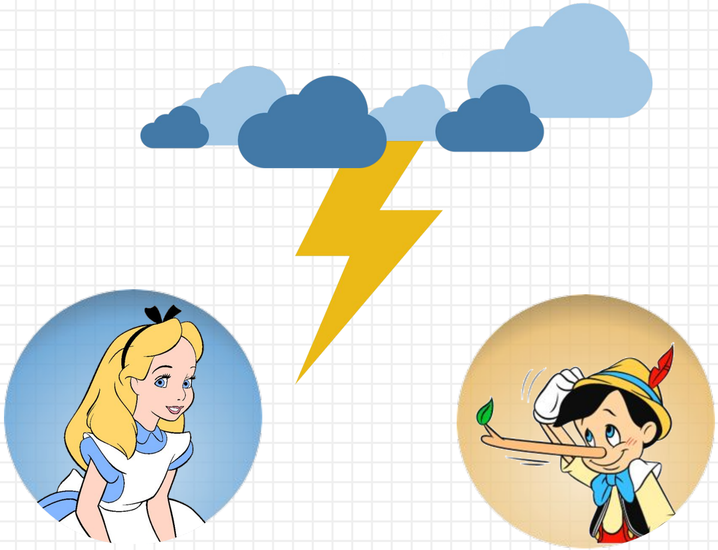
**[INPP21] Malicious securitY for Oblivious Polynomial Evaluation**

Malika Izabachène, Anca Nitulescu, Paola de Perthuis, David Pointcheval

- ✘ SNARK for Inner-Product over ciphertexts: adds security against malicious parties
- ✘ Reduce communication in 2PC with FHE
- ✘ Applications to PSI



# Conclusions

# Conclusions

## Quadratic Programs and SNARKs over fields

- ✘ Lots of implementations, but they fall short in one aspect
- ✘ Emulating ring arithmetic on SNARKs is expensive and unfriendly to applications
- ✘ Today's cost: Compilation to circuits over fields, costly preprocessing

## Rinocchio: SNARKs for Ring Arithmetics

- ✘ Circuit-SAT for arithmetic circuits over commutative rings: Quadratic Ring Programs (QRP)
- ✘ Better fits FHE schemes arithmetics $\mathbb{R}_q = \mathbb{Z}_q[x]/R(x)$ even for $q$ not prime
- ✘ Supports sub-circuits for special operations in FHE: modulo switching
- ✘ Designated Verifier only
- ✘ Can be turned Zero-Knowledge using Context Hiding techniques

# Conclusions

## Quadratic Programs and SNARKs over fields

- ✘ Lots of implementations, but they fall short in one aspect
- ✘ Emulating ring arithmetic on SNARKs is expensive and unfriendly to applications
- ✘ Today's cost: Compilation to circuits over fields, costly prepocessing

## Rinocchio: SNARKs for Ring Arithmetics

- ✘ Circuit-SAT for arithmetic circuits over commutative rings: Quadratic Ring Programs (QRP)
- ✘ Better fits FHE schemes arithmetics $\mathbb{R}_q = \mathbb{Z}_q[x]/R(x)$ even for $q$ not prime
- ✘ Supports sub-circuits for special operations in FHE: modulo switching
- ✘ Designated Verifier only
- ✘ Can be turned Zero-Knowledge using Context Hiding techniques

## Open Questions

- ✘ Other Encodings over rings → publicly verifiable
- ✘ More efficient instantiations: Security assumptions over rings: L-O extractable vs PKE

# THANK YOU


eprint.iacr.org/2021/322

# Credits

Special thanks to all those who made and released these resources for free:

- ✘ Presentation template by SlidesCarnival
- ✘ Illustrations by Disneyclips, Iconfinder and Flaticon